Docket No.: OIN 1004-1US
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Bart A. Meltzer

Application No.: 09/173,858 | Confirmation No.: 4734

Filed: October 16, 1998 | Art Unit: 2178

For: DOCUMENTS FOR COMMERCE IN TRADING PARTNER NETWORKS AND INTERFACE DEFINITIONS BASED ON THE DOCUMENTS | Examiner: C. L. T. Huynh

COMPILATION OF EXHIBITS

This compilation of exhibits accompanies the declarations submitted herewith. Because of the extensive size of the exhibit documents, Applicants submit one copy of each of the following documents:

A. Tenenbaum, Jay M., Tripatinder S. Chowdhry and Kevin Hughes, "Eco System: An Internet Commerce Architecture" Computer May 1997: 48-55

B. Glushko, Robert J., Jay M. Tenenbaum, Bart Meltzer, "An XML Framework for Agent-based E-commerce" Communications of the ACM, Vol. 42, No. 3, pp. 106-109 & 111-114 (March 1999)

C. "index.html" from cbl/072 directory (file date stamped in 1997)

D. Selected files from cbl/072 directory (date stamped in 1997)

E. Selected files from cbl/075 directory (date stamped before January 21, 1998)

F. "Requirements and Tasks for the January Demo, (Updated 1/6/98 by Kenneth)", file named "demo_req_tasks.html" from Veo/web/dev/documents/old/demo directory (date stamped before January 21, 1997)

G. "imdesc.xml" from cbl/ingram/01 directory (date stamped before January 21, 1997)

H. Selected files from cbl/ingram/01 directory (date stamped before January 21, 1997)

I. Glushko, Robert J., Implementing Domain-specific Commerce Languages with a Common Business Library, Slides 29-31 (delivered July 25, 1998) accessed at http://groups.haas.berkeley.edu/citm/conferences/cec/Presentations/Session3/glushko.pdf on October 26, 2006

J. Excerpts from W3C "note" WSDL version 1.1 (March 15, 2001) accessed at http://www.w3.org/TR/wsdl

Exhibit A. Tenenbaum, Jay M., Tripatinder S. Chowdhry and Kevin Hughes, "Eco System: An Internet Commerce Architecture" Computer May 1997: 48-55

Cover Feature

Eco System: An Internet Commerce Architecture

Robust electronic commerce will require several proprietary systems to interoperate. CommerceNet is proposing a framework of frameworks that will bridge among conflicting platform requirements.

Jay M. Tenenbaum

Tripatinder S. Chowdhry

Kevin Hughes
CommerceNet

The Internet is revolutionizing commerce. It provides the first affordable and secure way to link people and computers spontaneously across organizational boundaries. This is spawning numerous innovative enterprises—virtual companies, markets, and trading communities.

But the Internet's potential is imperiled by the rising specter of digital anarchy: closed markets that cannot use each other's services; incompatible applications and frameworks that cannot interoperate or build upon each other; and an array of security and payment options that confuses consumers.

One solution to these problems is an object-oriented architectural framework for Internet commerce. Several major vendors of electronic-commerce solutions have announced proprietary versions of such a framework. The major platforms are

- IBM CommercePoint
- Microsoft Internet Commerce Framework
- Netscape ONE (Open Network Environment)
- Oracle NCA (Network Computing Architecture)
- Sun/Javasoft JECF (Java Electronic Commerce Framework).

Recently, four of these companies have agreed to support a common distributed object model based on CORBA IIOP (Common Object Request Broker Architecture Internet InterORB Protocol). Yet for commerce on the Internet to thrive, such systems must also interoperate at a business application level. (For more information see the "Major E-Commerce Platforms" sidebar.) A consumer or business using one framework should be able to shop for, purchase, and pay for goods and services offered on a different framework. This is currently not possible.

In response, CommerceNet is organizing Eco System, a cross-industry effort to build a framework of frameworks, involving both e-commerce vendors and end users. This project is challenging from a technical perspective because information technology is moving so fast that there's seldom time for even de facto standards to emerge. Instead, we must deal with de facto interoperation—making incompatible products already in the marketplace communicate. Our philosophy is simple: Protocols, formats, and the like should not hinder business.

The success of this process clearly depends on market leaders in each area participating actively on their respective task forces. Admittedly, in past battles for market dominance (such as in operating systems and desktop PCs), it was difficult to bring leading players to the table. For robust Internet commerce, however, interoperability is so fundamental that we have to turn the concept of openness on its head—it's not just publishing an API. Everyone's software has to work together because no single company can control what platform its customers will use.

OVERVIEW

As proposed, Eco System will consist of an extensible object-oriented framework (class libraries, application programming interfaces, and shared services) from which developers can assemble applications quickly from existing components. These applications could subsequently be reused in other applications.

We are also developing a Common Business Language (CBL) that lets application agents communicate using messages and objects that model communications in the real business world. A network services architecture (protocols, APIs, and data formats) will insulate application agents from each other and from platform dependencies, while facilitating their interoperation.

Functionally, Eco System fills three distinct roles. It is

0018-9162/97/$10.00 © 1997 IEEE

- a layer of middleware that facilitates agent interoperation through services such as authentication, billing, payment, and directories;
- an object-oriented development environment that encourages the reuse of e-commerce modules (even modules that represent the product line of an entire company); and
- an industry roadmap and interoperability example that promotes open standards and helps technology vendors communicate with end users about product features.

A framework of frameworks

In object-oriented parlance, a framework is an almost complete application that users can customize or extend to address particular needs. Eco System is a framework for building Internet markets. Specifically, it's a framework of frameworks that model key business processes and services. Because frameworks build on each other, the resulting applications are tightly linked through a shared-services infrastructure. Eco System's frameworks fall into four general categories, as Figure 1 shows.

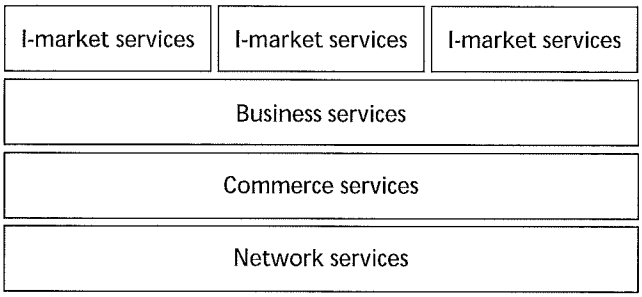

Figure 1. Four general categories of Eco System frameworks.

- I-*market services* are those that serve an Internet market. These are vertical markets of closely aligned businesses. Examples are real estate (title search, loan, and escrow services), securities trading (buy, sell, and quote services), or any vertical supply chain ("solicit bid," "issue request for quote," and "issue purchase order" services).
- Business services include generic business processes and applications common to multiple I-markets. These include retail (shopping, order fulfillment, and shipping) and business-to-business (procurement, order entry, inventory and supply chain management, and logistics) functions. Vendors may have initially developed such services for a specific I-market and later general-

Major E-Commerce Platforms

IBM's CommercePoint, a suite of e-commerce services, attempts to provide end-to-end business solutions (http://www.internet.ibm.com/commercepoint). It includes software packages for electronic storefronts (including credit card transactions using SET and back-office functions), purchasing (requests for proposals, electronic data interchange, and bidding), and distribution.

Netscape ONE (Open Network Environment) is a platform-independent, network-centric application development environment based on publicly defined open standards (http://home.netscape.com/comprod/one/white_paper.html). Key technologies include HTML, Java and JavaScript 1.1, CORBA IIOP, and broad support for open communication and collaboration protocols (HTTP, NNTP, SMTP, IMAP4, and POP3) and security services (Secure Sockets Layer 3.0 and X.509v3). Applications interact through these interfaces (available on Netscape clients and servers), eliminating the sharp distinction between client- and server-side development.

Oracle's Network Computing Architecture (NCA) combines Web technology (HTTP and HTML) with CORBA 2.0 and IIOP to provide distributed computing in a networked environment. NCA also supports ActiveX/COM clients through open COM/CORBA interoperability specifications ratified by the Object Management Group. Key components include "pluggable" objects called cartridges that use IDL to identify themselves to other objects in a distributed system (see http://www.oracle.com/nca/html/nca_wp.html).

Sun and JavaSoft's Java Electronic Commerce Framework (JECF) is an open platform for purchasing, banking, and finance (http://www.javasoft.com/products/commerce). It provides a user interface (or wallet) for online purchasing and other financial transactions; a secure, encrypted wallet database; access to strong cryptography; applets; and a purchasing infrastructure. Java Cassettes implement specific online transaction protocols such as SET, Mondex, and CyberCash CyberCoin.

These four vendors announced this March that they would redesign their networking products to support CORBA. Moreover, they promised to deliver some of these CORBA-compliant versions as early as this month. They are also expected to endorse the use of Java Beans, a platform-independent, component-based software architecture based on Java (see http://splash.javasoft.com/beans/WhitePaper.html).

This leaves Microsoft, which uses its proprietary Distributed Component Object Model (DCOM) architecture, as the major non-CORBA-compliant holdout. DCOM is an OLE derivative for networks, which runs only on Windows and also uses Microsoft's proprietary ActiveX components. These technologies support Merchant Server, a Microsoft product that allows Internet service providers to offer electronic storefronts supporting SET for about $3,500 (see http://www.microsoft.com/merchant). Industry observers point out that Microsoft recently endorsed a Hewlett-Packard proposal to bridge the ActiveX and CORBA object models.

Although the companies supporting CORBA are CommerceNet members, Microsoft is not. This situation—in which the major market shareholder fails to participate—is common to similar industry consortium efforts. As CommerceNet's interoperability initiatives gain momentum, we hope that Microsoft will become an active participant.

Table 1. Sample service request messages.

Service	Message
Payment	Make a payment
	Obtain payment
	Use a credit card
	Have I been paid yet?
Shipping	Schedule a shipment
	Check the status
	Get a quote
Catalog	Perform a search
	Add, delete, or modify listing

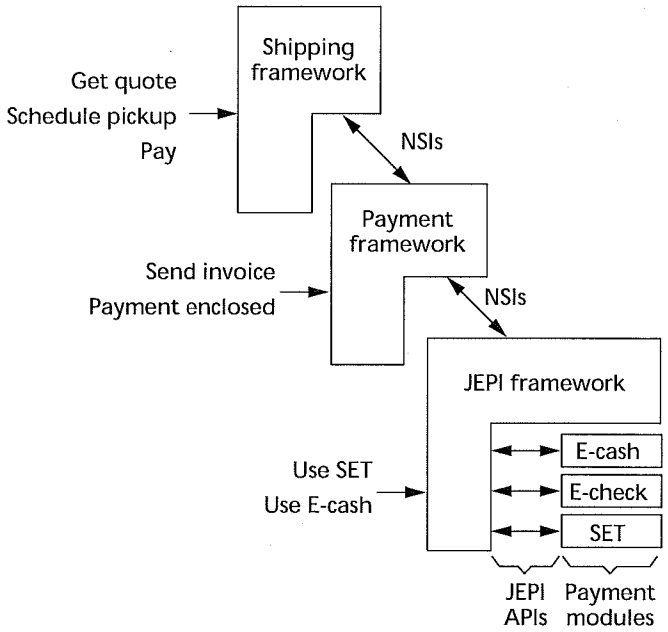

Figure 2. Frameworks communicate among themselves via NSIs and with application modules via APIs.

ized them for reuse. Marketware is a special subclass of these services that links buyers to sellers. (See the "Marketware" sidebar.)

- *Commerce services* are basic e-commerce services, such as digital "wallets," that allow individuals and companies to authenticate their identities, make payments, locate vendors, collaborate, and otherwise participate in an I-market. Advanced, next-generation commerce services will include secure multimedia mail, smart-card-based security and payment, digital-content delivery, application billing and accounting, transaction management, and agent management.
- *Network services* enhance the performance, reliability, and security of the Internet to accommodate mission-critical business needs. Examples include quality-of-service management, IP (Internet Protocol) multicast, delivery receipts, authenticated packets, and smart firewalls (those that pass packets only among authorized business partners).

Each framework specifies core services that all application objects belonging to that class (for example, payments and catalogs) must provide. They must also specify a network services interface (NSI). An NSI is a set of messages in an implementation-independent language (CORBA IDL, Interface Definition Language). These standard messages request services over a network and differ from APIs in that they are at a higher level and written in IDL. In addition, a framework must specify APIs for software modules involved in delivering services.

Services

Every application under Eco System—whether a catalog or an entire I-market—is a network-accessible service. Table 1 illustrates a few core services provided by three representative frameworks. The table lists paraphrasings of the NSI messages used to request the core services. These core services literally define what it means to be, for example, a payment, shipping, or catalog service. Vendors will differentiate their products by providing additional services beyond those specified in the framework. But the defining characteristic of a payment, shipping, or catalog object is its ability to respond to the minimal set of core service requests specified in the associated framework.

Modules can plug into frameworks via APIs; thus, some frameworks function as middleware, allowing access to several vendors' modules through a common set of requests. Object wrappers transform standalone and legacy applications (written before a relevant Eco service framework existed) into Eco services. Application modules plug into e-commerce platforms via APIs, and other applications can access them using standard NSI requests. The JEPI framework is an example of a payment platform. When fully developed, it will define standard APIs and protocols that allow interoperability of many incompatible payment solutions already on the market.

Figure 2 illustrates the hierarchical relationship of frameworks and the roles of NSIs and APIs.

GETTING FRAMEWORKS TO TALK

We are basing Eco System on CORBA 2.0, an emerging industry standard for distributed objects and networking. CORBA 2.0 includes the Internet InterORB Protocol (IIOP), which Netscape Communicator will support. Eco will also work with HTTP (hypertext transfer protocol), HTML (hypertext markup language), and Java. Figure 3 shows the Web-based architecture.

The following design decisions conform to emerging industry trends:

Marketware

A special class of Eco applications and services brings together buyers and sellers. Marketware is based on a common platform that developers can customize by plugging in different application modules. These modules serve as building blocks to implement a variety of value-added markets and market services:

- *Matchmaking* is a trading post where buyers and sellers can exchange goods or services. This service matches buyers and sellers on the basis of product descriptions and personal or company profiles (like, for example, Sun's Matchmaker).
- *Negotiation* services allow buyers and sellers to post offers specifying price ranges, quantities, delivery dates, and other terms. The service notifies parties in real time or via e-mail of close matches. Parties can respond by modifying their offers if so desired (as in, for example, the FastParts system).
- *Buy-sell brokering* allows buyers to post requests for quotations, which the service forwards to registered sellers with appropriate interest profiles. Sellers can respond with bids, which the service collects, sorts, and forwards to the buyer. (Shopping agents such as Andersen Consulting's BargainFinder are a special case of this service.)
- *Referrals and directory* services handle buyer requests for referrals. These services match requests against profiles of registered sellers using buyer-supplied criteria.
- *Aggregation* allows buyers to submit requests for goods and services, which the service pools with similar requests to obtain quantity discounts.

The marketware framework supports these applications by providing a common set of structures and functions.

- Standard profiles for buyers, sellers, and intermediaries. Profiles provide the information needed for a party to participate in market transactions. This information could include size and type of business, location and street address, terms, conditions, contracts supported, certificate information, credentials, credit rating, and references.
- Standard taxonomies of goods and services would allow parties to target particular transactions and filter out others. Taxonomies would use standard commercial classifications such as SIC (standard industrial classification) codes as well as custom ones. For example, a three-level hierarchy would classify products by industry (for example, computer), subarea (peripherals), and type (disk drives). CommerceNet is working to develop an evolvable "Taxonomy of Everything" for products.
- Standard CBL commands to invoke market actions such as buy, sell, bid, post request for quote, and locate interested buyers or qualified vendors.
- Authentication and authorization functions that use buyer and seller profiles to control what information a party can see or modify.
- Accounting and reporting of transactions for buyers, sellers, and market administrators.
- A notification service allows buyers and sellers to register their interest in selected market events (a new-bid posting, for example) and receive a CBL notification message when they occur.

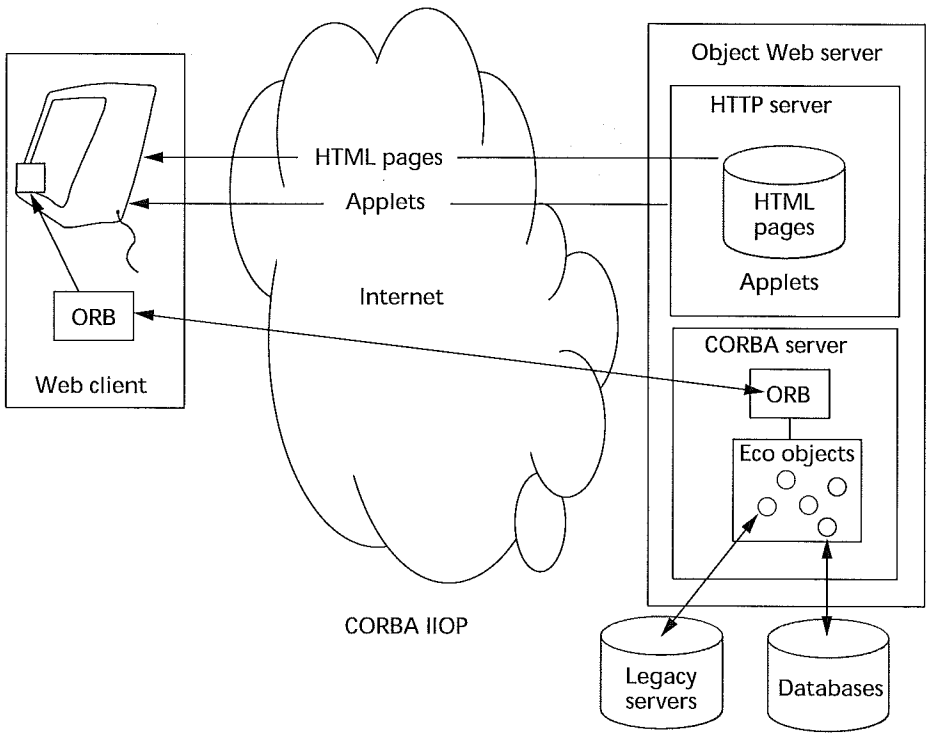

Figure 3. Eco services will be available as objects accessible via CBL commands sent over IIOP or HTTP/HTML sent by a browser. The architecture also incorporates Java applets, which link Web services to more robust transaction-oriented services via IIOP.

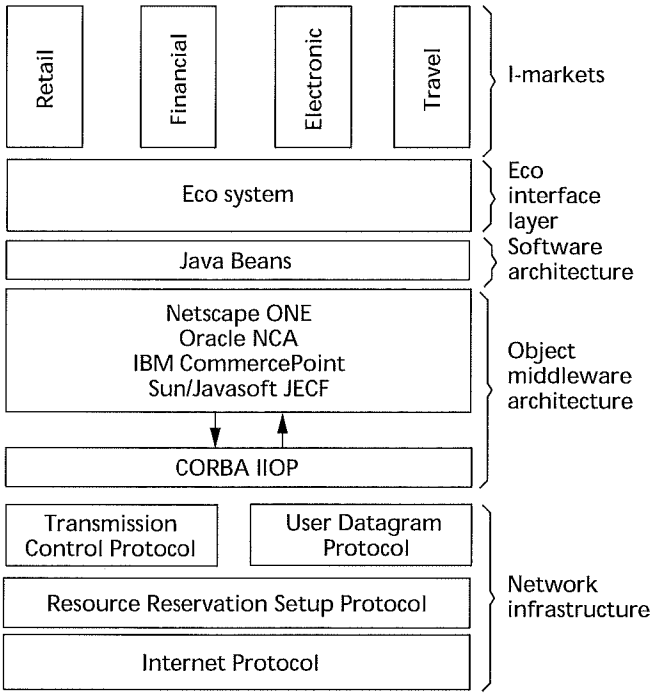

Figure 4. Protocol stack.

- *Network services.* Every Eco application will be a network-accessible service provided by agents.
- *Object Web.* Eco agents respond to CBL messages from other agents and to HTTP requests from browsers.
- *Industry compatibility.* As currently planned, Eco will foster interoperability among four of the five major e-commerce frameworks.
- *De facto interoperation.* Eco focuses on interoperation rather than standards. It will achieve interoperation in many ways, including the use of de facto standards implemented in Java and IIOP to achieve platform independence. Protocol negotiation, gateways, and mediators will provide semantic interoperation.
- *Scaleable, interchangeable building blocks.* Agents can direct CBL commands to a business, several businesses that have linked their catalogs or processes, a market (comprised of many companies), or a third-party intermediary.
- *Transparent outsourcing.* Eco will facilitate the outsourcing of business processes such as fulfillment, shipping, and payment processing.

Object orientation

Every Eco System service is a network-accessible object. As shown in Figure 3, objects respond to agents using CBL commands delivered over IIOP and to browsers using HTTP, HTML, and Java. This duality maintains compatibility with current Web sites and affords a graceful migration path. It's also compatible with emerging industry trends and anticipates the possibility that the next generation of HTTP and IIOP may someday merge. If the industry does not widely accept CORBA, agents will still be able to access the Web by using embedded semantic markup. Such embedded markup will let agents understand and respond to the information depicted graphically in a Web page. Microsoft and Netscape recently endorsed XML (Extended Markup Language), a simplified version of SGML used for embedding tags into HTML.

As shown in Figure 4, Eco imposes a layer of middleware on top of leading Internet commerce platforms such as Netscape ONE and Oracle NCA. It uses the CORBA IIOP architecture supported by these platforms and extends it to accommodate CBL agents.

Object bus. In CORBA, all objects connect to a common object bus, as shown in Figure 5. Thus, although we often depict Eco services hierarchically as in Figure 1, their actual implementation is flat; any Eco object can request a service from any other. This is convenient because situations do frequently arise where objects lower in the hierarchy require services from

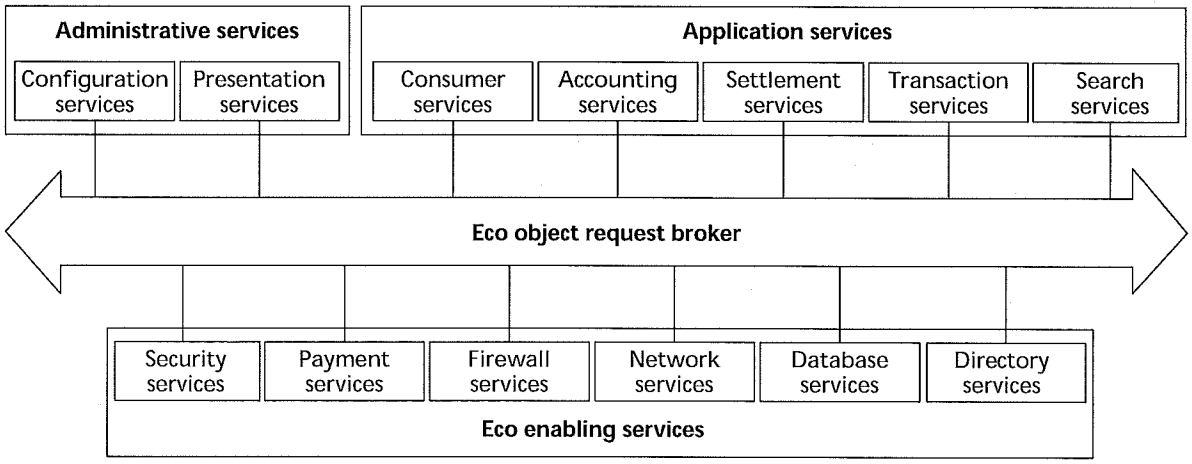

Figure 5. ECo object request broker acts as a bus between object-encapsulated services.

those above. For example, premium network services such as quality-of-service management or IP multicast may involve payments. Or a fulfillment service may need transportation I-market services.

IDL. CORBA and IIOP insulate application developers from most implementation and runtime details. CORBA provides IDL, a neutral definition language not tied to any specific programming language. Compiling the IDL generates object-oriented code implementing APIs. This allows any vendor to provide application object(s) that actually implement the specification. Vendors can write such objects in any language, and the objects can reside on any Internet-connected host. This architecture accommodates legacy applications by encapsulating them in an object wrapper and creating a corresponding IDL file as an interface. CORBA standardizes a CORBA-IDL-to-C++ mapping. JavaSoft and the OMG (Object Management Group) have released Java IDL alpha 2.2 for mapping IDL to Java.

Java. Object orientation allows developers to more quickly write and/or reuse applications to support changing business environments. Maintaining Eco's object orientation requires the use of an object-oriented language; CommerceNet has selected Java for this task.

Java is an interpreted language developed specifically with heterogeneous distributed networks and applications in mind. Vendor-neutral bytecode can be securely downloaded from the network as an applet that runs on a virtual machine residing on the user's system, most likely a Java-enabled browser. The Java runtime has built-in security features such as a bytecode verifier that enforces the Java security model (for example, disallowing pointers) and prevents malicious code from escaping the Java virtual machine (or "sandbox") and accessing the underlying operating system. Finally, Java Beans provides an architecture and platform-neutral API for creating and using dynamic Java components. Developers will be able to use a variety of development tools to assemble custom applications. These applications can draw on a rich variety of support services (such as event handling and persistence) that make Java Beans fully portable.

Protocol negotiation, mediators, and gateways

Application vendors are usually much more willing to agree on a metaprotocol than a standard. That's because a standard would require most to abandon rival technologies in which they have a substantial investment. Since today's computers can support multiple protocols, negotiation is a practical way of realizing de facto interoperation.

Negotiation protocols, bridging gateways, and mediators (smart gateways) have a part in accomplishing interoperability. Often, an application may not care what protocol it uses: "Just tell me what protocol you prefer, and I'll accommodate it if I can." This is the basic philosophy underlying the JEPI payments framework (see the "Payment Inter-operability" sidebar). In JEPI, sellers provide buyers with a list of payment types they accept (analogous to merchants displaying credit card logos in their store windows). Buyers then select the form of payment they wish to use, which implicitly

Payment Interoperability

In December 1995, the World Wide Web Consortium (W3C) and CommerceNet cosponsored the Joint Electronic Payment Initiative (JEPI) to bring key industry players together (CyberCash, IBM, Microsoft, Xerox, and British Telecom, among others) to ensure that multiple payment instruments, protocols, and transports will interoperate over the Internet.

JEPI is a metaprotocol built on top of two new Web protocols—PEP (Protocol Extension Protocol) and UPP (Universal Payment Preamble)—that let clients and merchant servers negotiate among and select payment mechanisms. Clients and servers can ask each other what forms of payment they support and negotiate a mutually acceptable payment mechanism.

PEP is a protocol for extending HTTP so that it can dynamically deploy applications that require more facilities than those provided by HTTP's request-response model. PEP associates new extensions to HTTP with a URL and uses a new `Protocol:` header field to carry the extension identifier and other necessary information—including possibly an implementation of the extension—between clients and servers. Like Java's protocol handlers, PEP provides the capability to automatically and dynamically download software component interfaces, enabling sophisticated applications such as distributed authoring tools to interoperate over the Web. PEP has been submitted to the IETF for inclusion in HTTP.

Don Eastlake built the Universal Payment Preamble on top of PEP. UPP is intended to provide a minimal layer that lets customers use a multipayment wallet and easily move from payment to payment. It provides a uniform vocabulary and syntax for naming options common to many payment systems, enabling clients and servers to exchange the necessary information and enter a specific payment system. This approach redefines each proprietary payment system as an URL-identified, PEP protocol extension implemented by a generic UPP protocol and module.

UPP negotiations occur via exchange of PEP protocol headers before or during shopping. Negotiation requests available payment choices, presents multiple choices, demands or makes a selection, and accepts or rejects choices. The payment protocol guarantees security, not UPP.

JEPI completed phase 1 in April 1997 with a demonstration at the Sixth International World Wide Web Conference of a JEPI implementation comprising two payment instruments, CyberCash and GCTech's GlobeID. W3C met with its members at that meeting to consider phase 2 strategies, which may include: validation/revision of UPP/PEP (JEPI used the August 1996 version of PEP, which was subsequently revised for consideration by IETF); incorporation of more payment systems (SET and micropayments), smart card integration; wallet and cash register APIs; and extension of HTML for micropayments. CommerceNet has committed to phase 2 development, according to CommerceNet's Jim Galvin, project manager for JEPI. For more information, see Eui-Suk Chung and Daniel Dardailler's "White Paper: Joint Electronic Payment Initiative (JEPI)," http:/www.w3.org/pub/WWW/Payments/white-paper.html.

AIMSNet

Ram Sriram

AIMSNet, a product of the Agile Infrastructure for Manufacturing Systems (AIMS) program, is a working example of an I-market in the making. Using AIMSNet, an intercompany network (using the Internet) links companies like Lockheed Martin and its suppliers, allowing multicompany project teams to exchange technical and business information, collaborate on design, post quotes and purchase orders, tender or accept bids, find potential suppliers and partners and track project milestones. More than 10 companies currently use AIMSNet, and dozens more are joining soon.

One of AIMSNet's powerful features is support for collaborative design. Its Multimedia Environment for Collaborative Engineering (MECE) is an online, shared notebook system developed by Lockheed Martin. This allows project team members to assemble and share information, such as design rationale and program decisions, in the form of text, audio, video, and screen snapshots. It also accommodates 3D design and manufacturing information by using VRML. VRML provides a 3D model independent of any specific CAD program. Team members use these tools to review information, collect comments, and make recommendations and changes. Current AIMSNet users are large programs within aerospace companies that develop complex systems such as satellites, rocket engines, missiles, and so on.

This effort, funded by the US Defense Advanced Research Projects Agency, has also developed templates to standardize transactions between companies. An important e-commerce concept, templates convey information between companies in a standard format easily accessed from anywhere through a HTML browser. Users can import information from legacy systems as well as through industry standard protocols. These templates also serve as simple front-end-to-remote databases that are network accessible.

Work is in progress to provide multitier supply chain coordination and facilities for evaluating and selecting suppliers. The coordination agent enables team members to track events critical to a project's success. The agent filters, sorts, prioritizes, and presents status information coming from various sources to project members based on their requirements. This helps project members manage the project from their own perspective. The supplier selection agent provides a mechanism for rapidly identifying key partners that can meet a project's multiple criteria. AIMSNet currently offers users a preliminary version of these services.

AIMSNet, an industrial commerce infrastructure, is currently piloted as an aerospace I-market but can be easily customized to several other I-markets including automotive, electronics, and construction.

Ram Sriram is AIMS Program Director for Lockheed Martin Missiles & Space. Contact him at sriram@aic.lockheed.com.

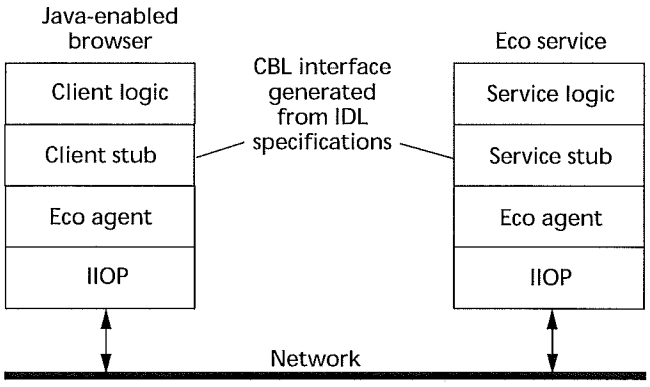

Figure 6. IDL provides a neutral definition language for connecting distributed applications (through CBL and Eco agents) in a platform-independent manner.

selects the appropriate protocol (SET or Mondex, for example).

An alternative to protocol negotiation is simply to translate between proprietary protocols using a gateway service. Gateways work well with functionally similar protocols that differ in syntactic details. Thus, gateways are often a good way for legacy database applications to communicate (for example, my SAP purchasing system can talk to your Oracle order-entry system) because the applications involved are reasonably well standardized at a functional level.

Gateways can also complement protocol negotiation. Namely, one alternative can be for each party to adhere to their favorite protocol and employ gateway services. In effect, the parties agree to disagree.

Mediators are smart gateways, which can negotiate a mutually acceptable protocol with each of several sites, retrieve information from each site, and integrate it. Mediators were originally developed for advanced information retrieval tasks, but are well-suited to e-commerce tasks such as integrating the catalogs and business systems of several cooperating firms.

Common Business Language

NSI messages, business objects, and product taxonomies will constitute a CBL for Internet commerce. Eco extends IIOP by adding two new levels of abstraction: CBL messages and CBL agents. A CBL message is an object-oriented alternative to the ad hoc text strings currently used in electronic data interchange. Each framework inherits the service requests and business objects of those frameworks upon which it builds, specializing and extending the inherited entities to provide new functions.

CBL agents provide a baseline set of common (Telescript-like) services that all e-commerce applications can build on. They include basic authentication,

EDI Interoperability Testing

EDIINT, the Internet Engineering Task Force (IETF) workgroup, has recommended standards for interoperable, secure electronic data interchange over the Internet. CommerceNet is sponsoring interoperability testing of these EDIINT recommendations among implementations from 10 vendors. These vendors include Actra Business Systems (a Netscape/GEIS company), Atlas Products International, AT&T, CyberPath, DaNet, Digital Equipment Corp., EDS, Harbinger, Premenos, and Sterling Commerce.

The vendors are checking the ability of their products to pass EDI securely among themselves. In January, five vendors demonstrated the successful exchange of digitally signed data among their products. This demonstration involved passing electronic documents over SMTP (Simple Mail Transport Protocol) using S/MIME (Secure Multipurpose Internet Mail Extension) encoding. Although the products all implement the S/MIME standard, factors such as certificate version differences and S/MIME support for multipart/signed documents still caused short-term interoperability problems.

Rik Drummond is chair of the IEFT working group, manager of the testing, and principal of the Drummond Group, an e-commerce consultancy. He anticipates the results of the EDIINT recommendations and the assurance of the CommerceNet interoperability testing to result in several secure, interoperable, off-the-shelf Internet EDI products in the next few months. The first group of five vendors will complete the total testing—exchanges of certificates, encrypted and signed messages, and signed receipts—by mid-May. A signed receipt is the basic mechanism for nonrepudiaton of receipt. In addition, the IETF is reviewing two draft standards—"MIME-Based Secure EDI" and "EDIINT Functional Specifications"—that outline the basis for secure, interoperable, Internet EDI. These standards set forth functional requirements for encryption, key management, content integrity, authentication, receipts, and tracking and error handling. They also recommend existing standards that fulfill these requirements. Both documents are available at http://www.ietf.org/ids.by.wg/ediint.html. Drummond expects these drafts to be accepted as Requests for Comments within the next few months. For more information about either the CommerceNet interoperability testing or the standards, contact him at drummond@onramp.net.

authorization, billing and accounting, micropayment, and directory services. We will base these agents on several lightweight agent architectures developed for use with Java, including IBM's Aglets and Mitsubishi's Concordia.

Eco's agent platform, depicted in Figure 6, provides an agent transport protocol and associated management and support services (creating and destroying agents, subcontracting tasks, delegating permissions and resources, and administering offers to buy or sell services). Using IDL, the CBL stub translates CBL messages into object requests to use IIOP-provided interoperability services.

PROJECT STATUS

In addition to the four major platform vendors, other organizations are active CommerceNet participants—Actra, Bank of America, Visigenic, the World Wide Web Consortium, and NIST, to name a few. CommerceNet recently agreed to cooperate with five Japanese organizations—NTT, the Japan Research Institute, Mitsubishi, NEC, and Oki—in developing functional prototypes of I-markets for a mall of malls and auto parts procurement. Additional I-market pilot programs include those for real estate and aerospace. The latter is already a working Internet-based network for manufacturing procurement; see the "AIMSNet" sidebar.

Another area in which CommerceNet is making a significant impact is in establishing standards and testing for secure electronic data interchange (see the "EDI Interoperability Testing" sidebar).

Although projects like AIMSNet allow pre-established trading partners to work together, we will use its results and EDI to create open I-markets in which an entire industry can come together for trade.

Internet commerce stands at a critical juncture. After an exhilarating start-up, further development hinges on bridging the chasm between early adopters and a true mass market. We envision Eco System as the foundation of that bridge.

Eco System is not just about creating an architectural framework of frameworks. It is, more importantly, about establishing an ongoing process and organization for achieving broad industry consensus on interoperability and reuse issues critical to open e-commerce. These issues are changing daily; visit http://www.commerce.net/Eco for the latest information. ❖

Jay M. Tenenbaum is founder and chair of CommerceNet, an industry association for e-commerce.

Tripatinder (Trip) S. Chowdhry, cofounder of the CommerceNet group, is the chief architect of Eco System. Chowdhry received his MBA from Kellogg Graduate School of Management and an MS in computer science from the University of Southern California.

Kevin Hughes is a consultant and CommerceNet Fellow.

Contact Tenenbaum at CommerceNet, 4005 Miranda Ave., Suite 175, Palo Alto, CA 94304; jmt@commerce.net.

Exhibit B. Glushko, Robert J., Jay M. Tenenbaum, Bart Meltzer, "An XML Framework for Agent-based E-commerce" Communications of the ACM, Vol. 42, No. 3, pp. 106-109 & 111-114 (March 1999)

ROBERT J. GLUSHKO, JAY M. TENENBAUM,
AND BART MELTZER

AN XML FRAMEWORK FOR Agent-based E-commerce

Emerging standards for commercial document exchange promise open business-to-business e-commerce.

COMMERCENET'S ECO SYSTEM INITIATIVE, LAUNCHED IN 1996, aims to transform the World-Wide Web into an agent-based infrastructure for Internet commerce.

Today's Web gives people unprecedented access to online information and services. But its information is delivered in format-oriented, handcrafted hypertext markup language (HTML), making it understandable only through human eyes. Software agents and search engines have difficulty using the information because it is not semantically encoded. Clever programmers work around some of HTML's inherent limitations by using proprietary tags or software that "scrapes" Web pages to extract content. Unfortunately, such ad hoc approaches do not scale. Proprietary tags require browser plug-ins, and scraping approaches require a customized script for each Web site. These approaches balkanize the Web, making it inaccessible to agents.

Tomorrow's Web will use the extensible markup language (XML) to encode information and services with meaningful structure and semantics that computers can readily understand. In Internet commerce, companies will use XML documents for publishing everything from product catalogs and airline schedules to stock reports and bank statements. They will also use XML forms to place orders, make reservations, and schedule shipments. Any agent with the proper authorization will be able to obtain computer-interpretable data sheets, price lists, and inventory reports through the Web or email, then request quotes, place orders, and track shipments.

By making the Web accessible to agents and other automated processes, XML will fundamentally transform the nature of e-commerce (see Maes et al.'s "Agents That Buy and Sell" in this issue). XML will eliminate the need for custom interfaces with every customer and supplier, allowing buyers to compare products across many vendors and catalog formats, and sellers to publish their catalog information once to reach many potential buyers. Online businesses will also be able to build on one another's published content and services to create innovative virtual companies, markets, and trading communities.

Web merchants might initially dread that XML-encoded information makes it too easy for buyers to compare prices and competitors to co-opt their content. But fear of lost business opportunity as e-commerce grows and the recognition that XML provides many other advantages for sellers (such as the ability to differentiate products in ways other than price) are likely to convince them to adopt richer markup formats. (see Wong et al.'s "Java-based Mobile Agents" in this issue). In time, most merchant Web sites will provide agent-searchable catalogs that supply product descriptions, as well as information about price and availability.

For consumers, the most obvious result of pervasive markup will be smart shopping agents that level the playing field in their dealings with sellers. Using Internet-wide shopping directories, these agents will be able to locate all merchants carrying a specific product or service, then query them in parallel to locate the best deals. Some merchants will provide sales agents that negotiate with shopping agents and generate customized offers in response to their solicitations. The shopping agents can then sort the offers they receive according to criteria set by their owners—the cheapest flight, the most convenient departure time, the roomiest aircraft, or some weighted combination. Cybermediaries will offer innovative brokering and referral services that match buying and selling agents, as well as order-aggregation services that increase their purchasing clout.

Agent-based shopping by consumers is just the tip of the e-commerce iceberg. Whenever a product is bought, information propagates back down the supply chain, triggering a series of distribution, manufacturing, and logistics events. Today much of this business-to-business information is exchanged through EDI messages. But traditional EDI is complex and expensive, because most messages travel over proprietary networks. Moreover, EDI's brittle syntax necessitates a custom integration solution between each pair of trading partners.

For these reasons, EDI transactions will increasingly take place over the Internet using an XML/EDI message format. Such messages will be more economical than traditional EDI messages, while being easier to validate and translate into the formats needed by applications at each end of the exchange [4]. This development will encourage businesses, including many that find traditional EDI too costly, to implement Web agents that respond to XML messages. This agent-based approach to enterprise integration is simpler and more open than traditional EDI, because it avoids the "pairwise tyranny" through which big companies impose proprietary message formats on small companies. More-

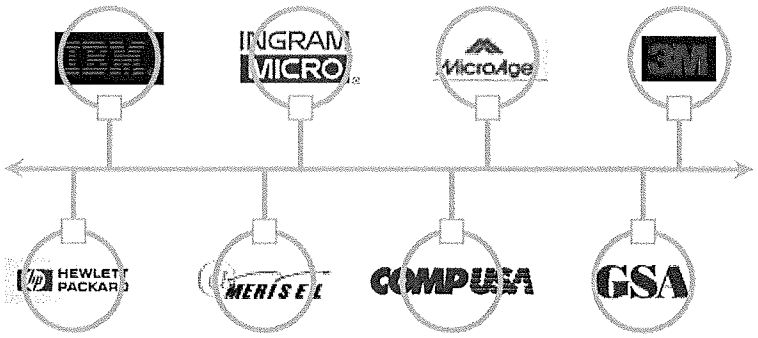

Figure 1. A supply Web linking PC manufacturers, distributors, and resellers

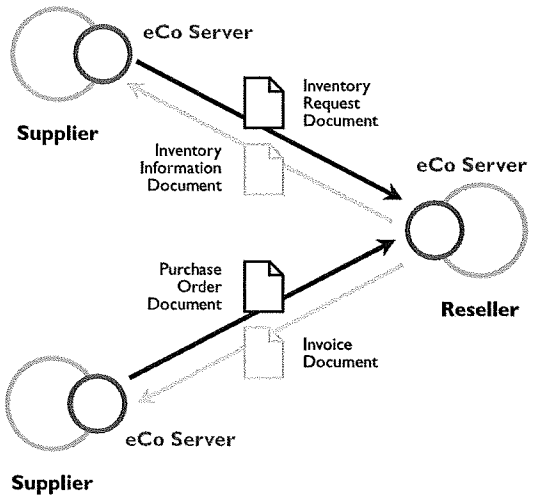

Figure 2. XML-based document exchange in the eCo System

over, publishing XML-encoded documents, such as data sheets and price lists, on the Web makes the information available instantly to all potential trading partners. Instant availability transforms rigid supply chains into "supply Webs," in which participants transact business spontaneously (see Figure 1).

The eCo System began as an architectural vision for open Internet commerce [5], proposed and evangelized by the 500-member worldwide CommerceNet Consortium in 1996. Conceived originally as a CORBA-based interoperability framework, the eCo System architecture was recast in 1997 on an XML foundation, due to XML's simplicity and widespread adoption by key vendors, including IBM, Microsoft, Netscape, and Sun.

Today's eCo System enables companies to communicate over the Internet using self-defining XML business documents that agents, as well as people, can easily understand. Business Interface Definitions (BIDs), posted on the Web, tell potential trading partners what online services a company offers and what documents to use when invoking those services. For example, a BID might allow a customer to order goods by submitting a purchase order or a supplier to check availability by downloading an inventory status report (see Figure 2).

A key element of the eCo System framework is the Common Business Library (CBL), an extensible, public collection of generic BIDs and document templates that companies can customize and assemble to go online quickly.[1] CBL includes XML message templates for the basic business forms used in ANSI X12 EDI transactions, as well as those used in such emerging Internet specifications as Open Trading Protocol (OTP) and Open Buying on the Internet (OBI). These specifications are mapped to each other using a dictionary of common business terms and data elements. A company can thus define its business interface in terms of any Internet standard mapped to CBL and communicate instantly with every other company that has done the same, even when the companies subscribe to different standards.

The eCo System framework overcomes two longstanding barriers to e-commerce. CBL facilitates spontaneous commerce between trading partners without custom integration or prior agreement on specific industrywide standards. And by being interpretable by both people and agents, XML documents provide an incremental path to business automation, whereby browser-based tasks are gradually transferred to computer agents. These advances eliminate much of the time, costs, and risks of traditional system integration. Moreover, the eCo System transforms closed trading partner networks into open markets and extends such enterprise applications as inventory management and production scheduling across entire supply chains.

XML is a simplified metalanguage, derived from SGML, emerging as the standard for self-describing data exchange in Internet applications. XML was developed by the World-Wide Web Consortium in 1997 and is being implemented rapidly by such major platform vendors as IBM, Microsoft, Netscape, and Sun Microsystems. XML's power

[1]The CBL was called the Common Business Language in earlier descriptions of eCo System. The change emphasizes CBL's function as a set of building blocks for XML applications and its role as a complement (rather than as a competitor) to ICE, OBI, OFX, OTP, RosettaNet, and other commerce languages.

derives from its extensibility and ubiquity. Anyone can invent new tags for particular subject areas, defining what they mean in document type definitions (DTDs). Content-oriented tagging enables a computer to understand the meaning of data, including, say, whether a number represents a price, a date, or a quantity.

This tagging significantly increases the functionality of Web e-commerce applications, because they can now do much more than simply display product data. For example, items in an XML-encoded catalog can be sorted by price, availability, and size.

One of eCo System's longstanding goals has been to enable businesses to build on one another's services to create virtual enterprises. Such plug-and-play commerce involves modeling enterprises as collections of services, some internal to a particular business, others provided by trading partners. Business services in eCo were originally defined as CORBA application programming interfaces (APIs). While the CORBA approach appears workable within organizations that control APIs, our experience in several prototypes suggests it is not practical for interenterprise integration. Fortunately, XML offers a promising alternative—agents interacting with business services through business documents.

Business documents represent a more intuitive and flexible way to access business services than programming APIs. It is much easier to interconnect companies in terms of the documents they exchange, on which they already largely agree, than in terms of their business system interfaces, which invariably differ. The coupling is looser, but loose coupling is better than no coupling at all.

XML's human readability is another significant advantage over CORBA. Just as HTML is a language for the eyes, CORBA is a language for CPUs, meant to convey information among programs, with no concession to human readability. XML documents are as readily interpretable by humans as they are by computers, especially with the aid of a style sheet [2].

Other proposals for agent languages suggest that first-order logic or other formal languages enable more precise specification of messages than XML [1, 3]. We prefer XML for two reasons—one language-theoretic, one practical. Expressing semantics in syntax rather than in first-order logic leads to a simpler evaluation function while needing no agreement on the associated ontologies. The practical argument, which is much more important for commercial success, is XML's ubiquity. The Web has made everyone appreciate the power of markup languages, practically assuring the widespread adoption of XML, as

Domain-specific Commerce Languages

The power of XML in enabling interoperability and simplifying the sharing and reuse of information between business domains is encouraging companies to work together to develop XML-based specifications for the business information they exchange most often. Sample specifications include:

- Open Trading Protocol. A consortium of banking, payment, and technology companies is specifying information requirements for payment, receipts, delivery, and customer support (www.otp.org). The goal of OTP is efficient exchange of information when the merchant, the payment handler, the deliverer of goods or services, and the provider of customer support are different entities with their own systems.
- XML/EDI. A group chartered jointly by CommerceNet, ANSI X12, and the Graphics Communication Association is defining how traditional X12 EDI business data elements should be represented using XML (www.xmledi.com).
- RosettaNet. This PC industry initiative is defining how to exchange PC product catalogs and transactions among manufacturers, distributors, and resellers (www.rosettanet.org).
- Open Buying on the Internet. The OBI initiative, launched by American Express and major buying and selling organizations, including Ford Motor and Office Depot, is automating large-scale corporate procurement of office and maintenance supplies (www.openbuy.org)
- Information and Content Exchange. CNET, News Corp., Vignette, and other information content providers are developing ways through ICE to create and manage networked relationships, such as syndicated publishing networks, Web superstores, and online reseller channels (www.w3.org/TR/1998/NOTE-ice-19981026).
- Open Financial Exchange. Originally proposed by CheckFree, Intuit, and Microsoft for the electronic exchange of financial statements among consumers, small businesses, and financial institutions, the OFX effort supports banking, bill payment, investment, and financial planning activities (www.ofx.net).

Share the Ontology in XML-based Trading Architectures

First bring semantic order to the world of XML.

Howard Smith and Kevin Poulter

Recent e-commerce application activity involving the extensible markup language (XML) has led to a proliferation of XML-based standards and markup language proposals. Among them are several designed to support site-to-site Web automation that lean naturally toward the agent paradigm of distributed computation.

Although XML represents a major step forward in e-commerce technology, business-to-business trading partners should also recognize XML's limitations. XML is not a cure-all for system interoperability, but a widely accepted foundation layer on which to build. Moreover, there are differing views on how to extend or complement XML to support agent-based e-commerce (see Glushko et al.'s "An XML Framework for Agent-based E-commerce" in this issue). This challenge is further complicated by debate over some fundamental questions: How should XML be extended to support the representation of business information? Should XML be enriched with tags reflecting higher-level concepts, especially business domains, such as standard business processes? How should foundation ontologies (from which higher-level content is composed) be defined? How can the numerous heterogeneous e-commerce frameworks (such as ICE, OBI, OTP, and XML/EDI) be unified to enable the expected low-friction market of the future? And will the future electronic marketplace be dominated by a series of commerce islands with trading groups isolated by the proprietary protocols and domain models with which their commerce agents interact?

Answers involve not only solving the related technology and intellectual challenges, but how to bring together the various communities of industrial standards developers. Each holds the essential elements of the overall solution. These communities, including EDI, Internet, knowledge engineering, and SGML, bring to the table subtly differing angles on the problem, including representation approaches associated with rich documents, publish/subscribe protocols, transactions, content syndication, and business semantics. To survive in this market, e-commerce component providers will have to support a number of different content formats and transaction frameworks, translating among them to achieve significant penetration. It appears that the main barrier to e-commerce lies in the need for applications to share information, not in the Internet's reliability and security.

Due to the wide range of enterprise and e-commerce systems being deployed by businesses and the way these systems are variously configured, the problem is particularly acute among large electronic trading groups. E-commerce will increasingly focus on trans-enterprise communication, while the number of trading partners and sophistication of e-commerce applications also increase. The need to unite business models, processes, and representation formats is greater than ever, while expectations run ever higher. Although many companies have already begun to organize, standardize, and stabilize their digital services in order to create and maintain sustainable network relationships with their trading partners, they are doing so only in conjunction with their immediate trading partners. This relatively narrow focus can limit the return on investment possible from each of these initiatives.

A global environment. There is now a need for e-commerce participants to create a global environment providing significant interoperability between the systems used by all engaged. Such an environment can be achieved through improved semantics within Internet transactions and in networked service definitions. It will facilitate consistent behavior among participants in large trading networks or within complex virtual organizations. Many of the foundation concepts needed to achieve this consistent behavior have already been established through work on distributed problem solving, intelligent agents, and knowledge sharing, yet to date these technologies have had little effect on Internet-based commerce.

Agent-based systems to support the next generation of Internet commerce must adopt common ontologies if they are to interact without misunderstanding. For example, content can be defined to enable application interoperation as well as information synthesis. An e-commerce standard being developed by major PC vendors, resellers, and distributors has shown by practical example in the PC distribution chain that quite sophisticated representation issues can complicate even straightforward commerce scenarios. For example, the required catalog model includes the need to represent the topology of the parts comprising a PC product.

But to bring semantic order to the world of XML, we have to be clear about what we mean by "ontology." The term is often used to refer to a vocabulary, yet even the terms within a simple vocabulary can be prone to misinterpretation, particularly in combination, unless they have been chosen carefully. Consider some of the problems already apparent in the plethora of e-commerce standards that

have emerged during the past few years. As new online trading environments are developed, the potential protocol mismatches between participants' commerce platforms can become major inhibitors to achieving industrywide e-commerce solutions and delivering supply-chain and market-efficiency benefits. Realizing Web automation in such complex environments reopens many of the problems and issues the knowledge-sharing and intelligent-agent communities have been wrestling with in such initiatives as the shared design environment, or SHADE, and the advanced technology operations system, or ATOS, using ontologies to enable agents working on different problems to interoperate over networks.

XML as a representation is just too forgiving at the document type definition (DTD) stage at the expense of the information processing stage. However, steps are being taken in the right direction; an example is the definition of schema languages to enable consistent schema semantics in the definition of objects in XML (such as by the World-Wide Web Consortium reflecting proposals from a number of organizations).

Consistent schema semantics will certainly enable efficient e-commerce using predefined DTDs between fixed networks of trading partners. But to enable the full benefits of agent-based e-commerce—where agents act in an autonomous or semiautonomous way, comparing and contrasting products or suppliers and negotiating with other agents—participating agents have to communicate in terms of a detailed ontology of the business domain.

The challenge for technology vendors, e-commerce participants, and standards bodies is to capitalize on the experience available in the knowledge representation and distributed agent communities.

Veo Systems is pursuing a pragmatic approach to solving some of these issues through the Common Business Library, an extensible, public collection of business interface definitions and document templates. This library is being rationalized and further developed by the CommerceNet eCo Framework Working Group established last year and should provide a foundation for addressing many of the unanswered questions in agent-based e-commerce. Ontologies will play a key role. C

HOWARD SMITH (howard.smith@ontology.org) is the director of Ontology.Org and a principal consultant (Internet) in Computer Sciences Corp. in Farnborough, Hampshire, U.K.
KEVIN POULTER (kevin.poulter@ontology.org) is chief technology officer of Ontology.Org and a principal consultant in Computer Sciences Corp., U.K.

HTML's heir apparent. XML may be theoretically less expressive than other formal languages, but we prefer a language that can be understood and produced by computer novices to a theoretically better one spoken only by computer scientists.

The significance of XML for integration extends beyond the Web to email, database records, and programming APIs. An XML parser imposes the same API on any XML data source, eliminating much of the need for custom programs to extract and integrate information from each source. So, integrating enterprise information from accounting, purchasing, manufacturing, shipping, and other functions can be accomplished by first converting each source to XML and then processing the parsed data stream. Put another way, each application need know only two source formats—its own and XML—rather than having to produce the native format of every other application.

XML by itself doesn't enable plug-and-play commerce. In addition to the language itself, a complete business integration solution also requires: standardized tags, or metadata, for each commerce community; a means for mapping between different metadata descriptions; and a server for processing XML documents and invoking appropriate applications and services. The eCo System framework starts with XML and adds these additional architectural and technology elements.

Specialized Markup Languages

XML makes it easy to create specialized markup languages that identify and describe buyers and sellers, the goods and services they want to buy or sell, and the various other document types involved in commerce. However, a vendor has obvious incentives for describing its offerings in ways that highlight its competitive advantages and that obscure comparison on features where it lacks an advantage. But if every business invented its own XML definitions for product catalogs, requests for quotes, price lists, purchase orders, invoices, transportation schedules, shipping notices, and delivery and payment receipts, the Web would be scarcely more usable as a platform for agents and other automated processes than it is today (see Smith's and Poulter's "The Role of Shared Ontology in XML-based Trading Architectures" in this issue).

Fortunately, many companies already recognize the need for information-exchange standards, uniting in several initiatives focusing on XML standards for particular industries or business

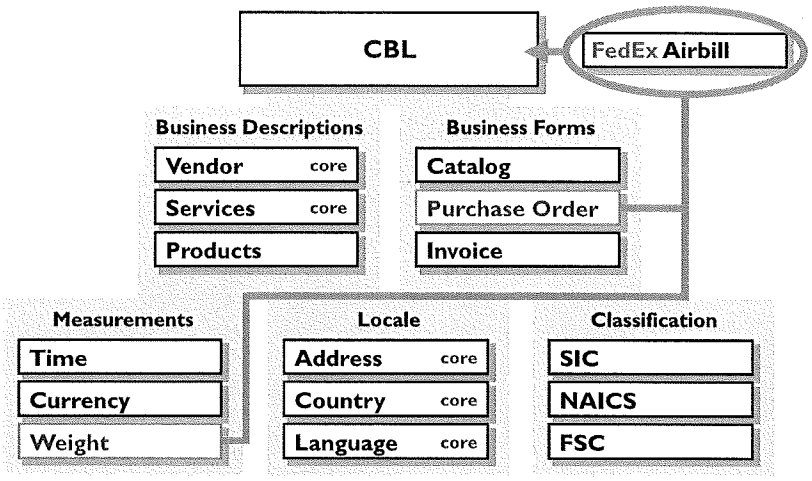

Figure 3. The Common Business Library

processes (see the sidebar "Domain-specific E-commerce Languages"). Unfortunately, these initiatives operate independently, doing little to facilitate interaction across industry and functional boundaries. The solution is to spur development of XML document models based on reusable semantic components common to many business domains. Such documents can be understood by any business through their common elements (such as address, date, and part number), while also providing a common mechanism for linking to the unique elements vendors need to differentiate themselves.

The CBL is designed to encourage development and use of generic XML document models. The library consists of information models for various concepts, including:

- Business descriptions, such as companies, services, and products;
- Business forms, such as catalogs, purchase orders, and invoices; and
- Standard measurements, such as date and time, location, and classification codes.

These models are represented as an extensible, public set of XML building blocks that companies can customize and assemble to develop XML applications quickly. Atomic CBL elements implement industry messaging standards and conventions, such as standard International Organization for Standardization (ISO) codes for countries, currencies, addresses, and time. Low-level CBL semantics are also derived through analysis of proposed metadata frameworks for Internet resources, such as the Dublin Core metadata element set developed by the Online Computer Library Center.

The next level of CBL elements use these building blocks to implement the basic business forms used in X12 EDI transactions, as well as those in OTP, OBI, and other emerging Internet standards.

A working group organized by CommerceNet and

```
<service>
<service.name>Order Service</service.name>
<service.location>www.veosystems.com/order</service.location>
<service.op>
<service.op.name>Submit Order</service.op.name>
<service.op.inputdoc>www.commerce.net/po.dtd</service.op.inputdoc>
<service.op.output>www.veosystems.com/invoice.dtd</service.op.outputdoc>
</service.op>
<service.op>
<service.op.name>Track Order</service.op.name>
<service.op.inputdoc>www.commerce.net/request.track.dtd<service.op.inputdoc>
<service.op.outputdoc>www.veosystems.com/response.track.dtd<service.op.outputdoc>
</service.op>
</service>
```

Figure 4. Fragment of an XML service definition for an eCo-compliant business application

other organizations recently began using CBL to create a base set of common terms, or mappings, between existing terms in commerce specifications, including OBI and OTP. The final result scheduled for release in mid-1999 will include a recommended base set of XML data elements, attributes, and definitions for use in e-commerce standards initiatives; they will be made freely available in public registries run by CommerceNet and other organizations. The Internet community, building on this foundation, will be encouraged to contribute additional elements and document models.

Agent-based shopping by consumers online is just the tip of the e-commerce iceberg.

Figure 3 shows how Federal Express might use CBL to create an XML version of its airbill by customizing a generic purchase order DTD with specific information about shipping weight. The generic purchase order, in turn, is assembled from more primitive CBL modules for address, date and time, currency, and vendor and product description. This example shows how reusing CBL components can significantly speed development of XML e-commerce applications and facilitate their interoperation.

When creating CBL, we found it helpful to extend XML with a schema language. The extensions add strong typing to XML elements so content can be readily validated. For example, an element called `CPU_clock_speed` can be defined as an integer with a set of valid values: {100, 133, 166, 200, 233, 266 Mhz}. The schema language also adds class-subclass hierarchies, so information is readily instantiated from class definitions. A laptop, for instance, can be described as a computer with additional tags for such features as display type and battery life. These and other extensions facilitate data entry, as well as automated translations between XML and traditional object-oriented and relational data models.

Trading partners not only have to agree on the meaning of message tags but understand how to use them for conducting business. In the eCo System, BIDs tell potential trading partners what online business services a company offers and which documents to use when invoking those services. In effect, services are defined by the documents they accept and produce. BIDs present a clean and stable interface to business partners, insulating them from a company's internal changes in technology, organization, and processes.

Figure 4 shows a fragment of a BID, defining an XML service for an eCo-compliant business. The service definition consists of two transactions—one for taking orders, one for tracking them. Each definition expresses a contract, or promise, to carry out a service if a valid request is submitted to the specified Web address. The order service requires an input document conforming to a standard `po.dtd` DTD in an industry registry operated by CommerceNet. If the service is able to fulfill the order, it returns a document conforming to a customized `invoice.dtd` whose definition is local. In effect, the company is promising to do business with anyone submitting a purchase order conforming to the XML specification it declares. No prior arrangement is needed.

A DTD is the formal specification, or grammar, for documents of a given type, describing the elements, their attributes, and the order in which they have to appear. For example, purchase orders typically include the names and addresses of the buyer and seller, a set of product descriptions, and associated terms and conditions, such as price and delivery dates. In the EDI world, the X12 850 specification is a commonly used model for purchase orders.

From Business Services to Virtual Enterprises

eCo servers provide the glue that links a set of internal and external business services to create a virtual enterprise or trading community. The server parses incoming documents and invokes the appropriate services (as specified by the applicable BID) by, say, handing off a request for product data to a catalog server or forwarding a purchase order to an enterprise resource planning system. The eCo server also handles translation tasks, mapping the information from one company's XML documents onto document formats used by its trading partners and into data formats required by its own legacy systems.

Following the service definition in Figure 4, when a company submits a purchase order, the XML parser in the eCo server uses the purchase order DTD `po.dtd` to transform the purchase order instance into a stream of information events. These events are then routed to any applications programmed to handle events of that type; in some

cases, the information is forwarded over the Internet to an entirely different business. In the purchase order example, information coming from the parser may be acted on by various applications:

- An order entry system processing the purchase order as a complete message;
- An enterprise resource planning system checking inventory for the products described in the purchase order;
- A customer database verifying or updating a customer's address;
- A shipping company system using the address information to schedule a delivery; and
- A bank system using credit card information to authorize a transaction.

However, what is most important in such processing is what is left out. Trading partners need agree only on the structure, content, and sequencing of the business documents they exchange, not on API details. How a document is processed and what actions result are strictly up to the business providing the service. This focus on commerce elevates enterprise integration from the system level to the business level.

A True Marketplace

eCo System's top-level goal is to transform the Web into a true marketplace by enabling spontaneous, peer-to-peer exchange of electronic business documents among all companies. This document-based approach replaces complex, expensive, and proprietary business integration solutions with one that is simple, affordable, and open.

The eCo architecture recognizes that a single dominant e-commerce standard is unlikely, even within a particular business community (and certainly not across communities). Rather, there will be many standards. CBL, in particular, is not a single standard but a collection of common business elements underlying all EDI and Internet commerce protocols. Its reusable components speed implementation of standards and facilitate interoperation by providing a common semantic framework. This approach to standards implementation and interoperation is fundamentally different from that taken historically by standards organizations and software vendors. It occupies an openness high ground embracing all the new competing standards being developed to take advantage of XML.

The eCo system framework and CBL are being evaluated in several of the standards initiatives listed in the sidebar on domain-specific commerce languages, as well as two major market trials sanctioned by CommerceNet:

- The U.S. General Services Agency (GSA). The largest buying organization in the U.S., GSA is creating catalog interoperability across numerous government agencies. Until now, the catalogs belonging to participating agencies were implemented as relational databases, as static files, or as catalog applications. An eCo server transforms each of these information sources into a standard catalog service that responds to CBL queries by outputting an XML data stream conforming to a common catalog schema. The integrated source catalogs can then be searched through specialized user interfaces developed by various participating technology vendors.
- RosettaNet. The RosettaNet consortium of PC manufacturers, resellers, and distributors is developing integration standards for the PC distribution channel; participants include Compaq Computer, CompUSA, Dell Computer, Hewlett-Packard, IBM, Ingram Micro, Merisel, Microsoft, and Tech Data.

The XML document models used in these initiatives are being rationalized to identify common semantic elements. These elements will be added to various public CBL repositories and made freely available (for more detail, visit www.commerce.net and www.veosystems.com). ■

References

1. Finin, T., Fritzson, R., McKay, D., and McEntire, R. KQML as an agent communication language. CIKM '94. In *Proceedings of the Third International Conference on Information and Knowledge Management*, 1994, pp. 456–463.
2. Fuchs, M. Domain-specific languages for ad hoc distributed applications. In *Proceedings of the Conference on Domain-Specific Languages*, 1997.
3. Kimbrough, S., and Moore, S. On automated message processing in electronic commerce and work support systems: Speech act theory and expressive felicity. *ACM Trans. Inf. Syst. 15*, 4 (Oct. 1997), 321–367.
4. Laplante, M. Making EDI accessible with XML. *EC.COM 4*, 2 (March 1998), 23–26.
5. Tenenbaum, J., Chowdhry, T., and Hughes, K. eCo System: An Internet commerce architecture. *Comput. 30*, 5 (May 1997), 48–55.

ROBERT J. GLUSHKO (glushko@veosystems.com) is director of information engineering at Veo Systems, Inc., in Mountain View, Calif.
JAY M. TENENBAUM (jmt@veosystems.com) is chairman and chief scientist of Veo Systems, Inc., in Mountain View, Calif.
BART MELTZER (bart@veosystems.com) is chief technology officer of Veo Systems, Inc., in Mountain View, Calif.

Exhibit C. "index.html" from cbl/072 directory (file date stamped in 1997)

Common Business Language (CBL)

Version 0.7.2

Terry Allen

5 December 1997

Purpose

The purpose of the Compound Business Architecture(TM)(SM), or CBA(TM)(SM), is to specify common semantics, syntax, and message packaging for information held by and exchanged between electronic commerce components. Its focus is on the functions and information that are common to all business domains. For the time being, the Compound Business Architecture is known as the Common Business Language (CBL).

Major Changes from the Previous Version (0.7.1)

- reworked the MIME specification again, reinstating `command.dtd` per popular demand.

Common Semantics

The semantics of CBL are to be drawn from international standards where appropriate.

Where appropriate, semantics are drawn from the UN/EDIFACT Basic Semantic Unit data dictionary and certain ISO and IETF standards. Modules constructed to date are:

- datetime.mod, for Date and Time
- measures.mod, for Weights and Measures
- currency.mod, for Currencies
- countrys.mod and addresso.mod, for Geographical Information
- commatts.mod, for XML attributes common to all modules
- ttlattri.mod, for XML attributes having to do with valid start and end dates and times for elements
- pointers.mod, mostly for pointers to typed information
- servprim.mod, for elements used in service description DTDs
- servmeta.dtd, for metainformation that servers keep about documents
- meta.mod, for metainformation elements
- price.mod, for Price Information
- shipment.mod, for Shipping Information

- shipnote.dtd, for Shipping Notice
- paymento.mod, for Payment Information
- paynoteo.dtd, for Payment Notice
- proddesc.mod, for a Simple Product Description
- markdesc.dtd, for Market Description
- markpart.dtd, for Market Participant Information
- catentry.dtd, for a Simple Catalogue Entry
- acatalog.dtd, for a Simple Full Catalogue
- shopcart.dtd, for a Shopping Cart
- transact.mod, to provide a mode-independent basis for
- order.dtd, for a Purchase Order, and
- invoiceo.dtd, for an Invoice
- taxonomy.mod, for Product Taxonomy, which may occur in a catalogue entry

In addition to these semantics of trade, we need semantics for the component-based services of our system, such as:

- servdesc.dtd, for a Service Description
- Directory (Registry) Profile, perhaps also Server Profile
- servmeta.dtd, for a Server Metadata about Documents
- rfq.dtd, for Request For Quote
- Request For Proposal
- ots.dtd, for Offer To Sell
- inventoy.dtd, for Inventory Info
- Authorization Information
- Trading Partner Agreement

To support message packaging and information discovery and exchange we add:

- manifest.dtd, for a Manifest for MIME Message Contents
- guide.dtd, for a Guide to Transaction Negotiation document that outlines the state of negotiation and the response expected from the other party
- semantic.dtd, for DTD Semantics
- cblcat.dtd, for URL-URN bookkeeping
- command.dtd, for commands to CBL servers
- modify.dtd, for describing modifications made to one documents in its revision
- infodesc.dtd, for a Request for Information
- response.dtd, Response to Request for Information

Finally, within each industry, those semantics specific to its domain must be defined. This work generally falls outside of the semantics of CBL, although it can build on CBL. The following modules have been constructed but not yet classified.

- schedule.dtd, for a Schedule for Events

Syntax

CBL semantics are to be encoded in XML DTDs (Document Type Definitions), suitably modularized so that the semantic primitives found in CBL are available to all the XML DTDs used in the system.

It may be convenient to make it an application convention of CBL that all instances must have as their root element the topmost element in the DTD to which they conform. This application could not be enforced by XML parsing, and would have to be enforced separately. No decision has been made on this point yet (a few DTDs have more than one top-level element, although it is intended to create more DTDs to eliminate that circumstance).

For the semantics of specific industries Commerce Type Definitions (CTDs) are needed; these are XML DTDs for specific industries, but fall outside of CBL.

CBL linking will employ, and if need be augment, the facilities of the XML Linking specification (XLL). As of the date of this document's writing, this specification was still unstable, and the details of the linking attributes in the CBL DTDs should be considered as a sketch.

Message Packaging

In addition to specifying semantics and syntax, CBL specifies a method of constructing messages out of multiple parts, and how these parts are packaged for secure delivery using MIME. At present we are developing a "Compounddoc" Type Parameter Media Subtype and "CBL 1.0" Schema Parameter Value for MIME Multipart/Related for this purpose. This format involves the use of an instance conforming to the manifest.dtd. A manifest points to the parts of the MIME message by their Content-IDs, and calls out the Document Entity, which for CBL must be an instance of `manifest.dtd`. The MIME-plus-manifest layer gives us a generic compound document wrapping mechanism for XML and SGML (specified separately and short-circuited here), with a defined constraint on content for our own purposes.

The `manifest.dtd` makes provision for the use of a catalogue that is not (yet) employed in CBL (and that may well not be the SGML Open catalogue). I envision that the complete list of all the pieces of the compound document will appear in the catalogue,

The Guide document is entirely specific to CBL. (See the Guide to the Guide DTD). It describes the compound document or compound document set formed by a transaction negotiation from the standpoint of CBL (and a human, thus this is a suitable view to use in an human interface such as the proposed VBBE). It looks as though a CBL MIME message would seldom end with a Guide document, and in my MIME examples I have included additional documents referred to by the Guide document. However, the Guide's references to other documents do not assume that they have been shipped along with the Guide.

For previous versions of CBL, I constructed documentation that has not been updated, although it may still be useful. For a scenario for registering information with a server see CBL Usage Scenario: Register Information. For a transaction, see CBL Usage Scenario: Simple Retail.

XML Design Strategy and Test Files

The `.dtd` modules invoke `.mod` modules and add element and attribute declarations to form schemas for XML documents useful in e-commerce. To combine them to form higher-lever schemas, we use DTDs specifying sets of links instead of inclusive doctypes (see transact.dtd for an example of this technique). These links bear attributes indicating whether the targets they point to are properly content of the parent document or lie outside of it.

Each XML test file invokes one of the .mod or .dtd modules, without further declarations (no internal subsets allowed!) except for module tests, where other modules may need to be declared.

The following test files have been used to exercise and validate (in the XML sense) the XML DTDs of CBL:

- qcblcato.xml
- qcomm.xml
- qentry.xml
- qfullcat.xml
- qg.xml
- qinfor.xml
- qinvoice.xml
- qinvetoy.xml
- qmani.xml
- qmark.xml
- qmod.xml
- qmdesc.xml
- qord.xml
- qots.xml
- qpay.xml
- qresp.xml
- qrfq.xml
- qsched.xml
- qsem.xml
- qservd.xml
- qservm.xml
- qship.xml
- qshop.xml
- qtaxo.xml

FPI for CBL

A possible Formal Public Identifier (FPI) for all of CBL would be

```
-//CNgroup//DTD CBL version 0.1//EN
```

and for individual pieces,

```
-//CNgroup//DTD CBL [module.name] version 0.1//EN
```

where **[module.name]** is something like addresso.mod or markpart.dtd.

Depending on the final state of the XML 1.0 specification, CBL DTDs may be referred to by means of URLs, FPIs, PIs (Public Identifiers), FPIs encoded as URNs, or URNs of some other sort.

Older Documentation for CBL Components

When CBL is complete, each component will have its own reference document (man page), formatted in HTML. For now, each component has a semantics file, with the same name as the component but ending in `.sem`. These files are XML instances conforming to semantic.dtd. For example, the `aggtrans.dtd` file is documented in the `aggtrans.sem` file. Every element type name, attribute name, and enumerated attribute value is defined in English. Where elements employ standardized semantics, the data types of their content and standardized attribute values are as specified in the applicable standard. Otherwise XML data typing has been eschewed except for enumerated lists of attribute values. As a result, except for certain obvious cases (a `urllink` attribute should have a URL as its value), element content and attribute values are specified only as strings. (These specifications not updated after 0.5.)

In addition, the following documentation is available:

- doctrans.htm, documenting transact.mod
- fixedvar.sem, documenting fixed attributes that appear with different values on different elements.
- currlist.sem, documenting currency code values; these are not specified in `currency.mod` because there are too many of them, but this data dictionary fragment can be used for data type checking after XML parsing.

CBL XML Design Preferences

It helps to have a list of design preferences (similar to a style sheet) for a set of DTDs in order to maintain consistency among them. For CBL DTDs the following design preferences are in force:

- Use lowercase where uppercase is not required by XML.
- Use periods for word dividers in NAME strings.
- Do not abbreviate words in NAME strings except for "info".
- Help avoid name space collisions by forcing enumerated lists into attribute value lists.
- Group like names so they alphabetize well, *e.g.,* `date.calendar` and `date.ordinal`.
- Permit variants of element content where ISO, etc., standards allow optional punctuation. Data content validators must be capable of handling these variations.
- No data attributes.
- No NOTATIONs.
- No marked sections, including CDATA marked sections.
- No embedding of scripts in comments.
- No internal subsets: the active DTD must be identifiable by a URL or URN, so all declarations must be external.
- No general entities aside from those defined by XML.
- Try to provide default values for attributes instead of using #IMPLIED, when there is an enumerated list.
- Follow the established naming conventions (should be listed) for element and attribute names, including strings "pointer", "set", and "attrib".
- Assign no FPIs or URNs without consultation.

For instances, we want to make it a convention that elements declared EMPTY in a DTD may be represented only by the `<element/>` syntax, and that elements that may have content but happen not to may be represented only by the `<element></element>` syntax.

CBL Design Rationale

The design of CBL's XML is heavily influenced by the author's common sense, life experience, and experience in writing and using SGML DTDs. The UN/EDIFACT information typology has been used for the purpose of unifying the semantics of terminal elements.

The MIME part of CBL is proposed in reaction to previous work on SGML and MIME in the IETF, and refines an earlier proposal by myself. I also learned much about MIME in the course of the OCLC's Metadata conference series, through discussion of Uniform Resource Characteristics in the IETF, and through a review of MIME specifications in November 1997.

Exhibit D. Selected files from cbl/072 directory (date stamped in 1997)

```
<!-- catentry.dtd   Version:  0.6 -->
<!-- Purpose:  provide simplest catalogue entry -->
<!-- Terry Allen  25 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % proddesc SYSTEM "proddesc.mod">
%proddesc;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ELEMENT catalogue.entry (meta?, market.participant.popular.name?,
   market.participant.info.pointer, catalogue.entry.id,
   product.description, max.quantity.per.customer?, stock.status?,
   quantity.in.stock?, shipment.set.pointer?,
   payment.method.set.pointer?, price.group,
   picture.pointer*, html.catalogue.entry.pointer?)>
<!ATTLIST catalogue.entry
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT catalogue.entry.id (%multilingual;)*>
<!ATTLIST catalogue.entry.id
   %common.attrib;
>

<!ELEMENT max.quantity.per.customer (#PCDATA)>
<!ATTLIST max.quantity.per.customer
   %common.attrib;
>

<!ELEMENT picture.pointer (xll.locator)>
<!ATTLIST picture.pointer
   %common.attrib;
   %xll.exlink.attrib;
   cblpointer  CDATA    #FIXED      "outside"
>
```

```
<!-- cblcat.dtd  Version:  0.6.1 -->
<!-- Purpose:  associate local filenames/URLs with URNs -->
<!-- Terry Allen  27 Nov 1997 -->
<!-- developed from storage.dtd, for which the FPI is
      "-//Palm Tree Books//DTD USB-Storage v0.1//EN" -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ELEMENT cblcat (meta?, cblcat.entry.pointer+)>
<!ATTLIST cblcat
   %common.attrib;
>

<!ELEMENT cblcat.entry.pointer (catalogue.filename,
      (%xll.or.urn;))>
<!ATTLIST cblcat.entry.pointer
   %common.attrib;
>

<!ELEMENT catalogue.filename (#PCDATA)>
<!ATTLIST catalogue.filename
   %common.attrib;
>
```

```
<!-- command.dtd   Version:  0.7.2 -->
<!-- Purpose:  group command information -->
<!-- Terry Allen  5 Dec 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % command
   "convey.info | register.document | unregister.document
   | register.service | unregister.service | request.info"
>

<!ELEMENT command.set (meta?, authorization.info.pointer*,
   (%command;), command.target)>
<!ATTLIST command.set
   %common.attrib;
>

<!ELEMENT convey.info EMPTY>
<!ATTLIST convey.info
   %common.attrib;
   %party.attrib;
>

<!ELEMENT register.document EMPTY>
<!ATTLIST register.document
   %common.attrib;
   %party.attrib;
>

<!ELEMENT unregister.document EMPTY>
<!ATTLIST unregister.document
   %common.attrib;
   %party.attrib;
>

<!ELEMENT register.service EMPTY>
<!ATTLIST register.service
   %common.attrib;
   %party.attrib;
>

<!ELEMENT unregister.service EMPTY>
<!ATTLIST unregister.service
   %common.attrib;
   %party.attrib;
>

<!ELEMENT request.info EMPTY>
<!ATTLIST request.info
```

```
   %common.attrib;
   %party.attrib;
>

<!ELEMENT command.target (%xll.or.urn;)>
<!ATTLIST command.target
   %common.attrib;
>
```

```
<!-- countrys.mod   Version:  0.22 -->
<!-- Purpose:  group country code tokens -->
<!-- Terry Allen  18 Oct 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % country.name.attrib
   "country.name (AD | AE | AF | AG | AI | AL | AM | AN
      | AO | AQ | AR | AS | AT | AU | AW | AZ
      | BA | BB | BD | BE | BF | BG | BH | BI
      | BJ | BM | BN | BO | BR | BS | BT | BV
      | BW | BY | BZ | CA | CC | CF | CG | CH
      | CI | CK | CL | CM | CN | CO | CR | CU
      | CV | CX | CY | CZ | DE | DJ | DK | DM
      | DO | DZ | EC | EE | EG | EH | ER | ES
      | ET | FI | FJ | FK | FM | FO | FR | FX
      | GA | GB | GD | GE | GF | GH | GI | GL
      | GM | GN | GP | GQ | GR | GS | GT | GU
      | GW | GY | HK | HM | HN | HR | HT | HU
      | ID | IE | IL | IN | IO | IQ | IR | IS
      | IT | JM | JO | JP | KE | KG | KH | KI
      | KM | KN | KP | KR | KW | KY | KZ | LA
      | LB | LC | LI | LK | LR | LS | LT | LU
      | LV | LY | MA | MC | MD | MG | MH | ML
      | MM | MN | MO | MP | MQ | MR | MS | MT
      | MU | MV | MW | MX | MY | MZ | NA | NC
      | NE | NF | NG | NI | NL | NO | NP | NR
      | NT | NU | NZ | OM | PA | PE | PF | PG
      | PH | PK | PL | PM | PN | PR | PT | PW
      | PY | QA | RE | RO | RU | RW | SA | SB
      | SC | SD | SE | SG | SH | SI | SJ | SK
      | SL | SM | SN | SO | SR | ST | SV | SY
      | SZ | TC | TD | TF | TG | TH | TJ | TK
      | TM | TN | TO | TP | TR | TT | TV | TW
      | TZ | UA | UG | UM | US | UY | UZ | VA
      | VC | VE | VG | VI | VN | VU | WF | WS
      | YE | YT | YU | ZA | ZM | ZR | ZW) #REQUIRED"
>
```

```
<!-- currency.mod   Version:  0.5 -->
<!-- Purpose:  group currency primitives -->
<!-- Terry Allen  24 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % currency.code.attrib
"currency.code CDATA #REQUIRED"
>

<!ELEMENT currency.amount (#PCDATA)>
<!ATTLIST currency.amount
   schema:edifact CDATA #FIXED "currency"
   %currency.code.attrib;
>

<!ELEMENT noncurrency.amount (#PCDATA)>
<!ATTLIST noncurrency.amount
   unit  CDATA #IMPLIED
   issuer   CDATA #IMPLIED
>
```

```
<!-- datetime.mod   Version:  0.6 -->
<!-- Purpose:  group date and time information primitives -->
<!-- Terry Allen  26 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->
<!-- checked against ISO 8601:1988(E) 26 Nov 1997 by Terry Allen -->

<!ELEMENT date (#PCDATA)>
<!ATTLIST date
   schema:edifact CDATA #FIXED "date"
>

<!ELEMENT date.calendar (#PCDATA)>
<!ATTLIST date.calendar
   schema:iso8601 CDATA #FIXED "3.3 date, calendar"
>

<!ELEMENT date.ordinal (#PCDATA)>
<!ATTLIST date.ordinal
   schema:iso8601 CDATA #FIXED "5.2.2 ordinal date"
>

<!ELEMENT time (#PCDATA)>
<!ATTLIST time
   schema:iso8601 CDATA #FIXED "5.3 time of the day"
>

<!ELEMENT utc (#PCDATA)>
<!ATTLIST utc
   schema:iso8601 CDATA #FIXED "5.3.3 coordinated universal time"
>

<!ELEMENT week (#PCDATA)>
<!ATTLIST week
   schema:iso8601 CDATA #FIXED "5.2.3 week"
>

<!ELEMENT week.and.day (#PCDATA)>
<!ATTLIST week.and.day
   schema:iso8601 CDATA #FIXED "5.2.3 date identified by
   calendar week and day numbers"
>

<!ELEMENT date.and.time (#PCDATA)>
<!ATTLIST date.and.time
   schema:edifact CDATA #FIXED "dateandtime"
   schema:iso8601 CDATA #FIXED      "5.4 combination of date and
   time of the day"
>

<!ELEMENT duration (#PCDATA)>
<!ATTLIST duration
   schema:edifact CDATA #FIXED "dateandtime"
   schema:iso8601 CDATA #FIXED      "5.5.3.2 duration of time"
>
```

```
<!-- guide.dtd   Version:  0.6.2 -->
<!-- Purpose:  group negotiation guide information -->
<!-- Terry Allen  29 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % negotiation.step.content "initiate | conclude | cancel
   | advance | modify | failure">

<!ENTITY % cbl.component
   "request.for.quote.component | request.for.info.component
   | response.component | market.description.component
   | catalogue.entry.component | catalogue.component
   | schedule.component | inventory.set.component
   | offer.to.sell.component
   | market.participant.info.component | shopping.cart.component
   | order.component | invoice.component
   | payment.notice.component | shipment.notice.component
   | service.description.component | server.metadata.component
   | confirm.component
   | acknowledge.previous.component | escape.component"
>

<!ELEMENT guide (meta?, negotiation.identifier*,
   authorization.info.pointer*,
   negotiation.step, process.model)>
<!ATTLIST guide
   %common.attrib;
>

<!ELEMENT negotiation.identifier (#PCDATA)>
<!ATTLIST negotiation.identifier
   %common.attrib;
   %party.attrib;
>

<!ELEMENT negotiation.step (%negotiation.step.content;)>
<!ATTLIST negotiation.step
   %common.attrib;
   %party.attrib;
>

<!ELEMENT initiate EMPTY>
<!ATTLIST initiate
   %common.attrib;
>

<!ELEMENT conclude EMPTY>
<!ATTLIST conclude
```

```
   %common.attrib;
>

<!ELEMENT advance EMPTY>
<!ATTLIST advance
   %common.attrib;
>

<!ELEMENT cancel EMPTY>
<!ATTLIST cancel
   %common.attrib;
>

<!ELEMENT failure EMPTY>
<!ATTLIST failure
   %common.attrib;
>

<!ELEMENT modify (modification.set.pointer*)>
<!ATTLIST modify
   %common.attrib;
   %party.attrib;
>

<!ELEMENT cbl.component.group (%cbl.component;)+>
<!ATTLIST cbl.component.group
   %common.attrib;
   %party.attrib;
>

<!ENTITY % cbl.component.or.group
   "%cbl.component; | cbl.component.group"
>

<!ELEMENT process.model ((%cbl.component.or.group;)*, we.are.here,
   (%cbl.component.or.group;)*)>
<!ATTLIST process.model
   %common.attrib;
>

<!ENTITY % component.attrib
   "schema.name (cbl | noncbl) 'cbl'"
>

<!ENTITY % cbl.component.contents
   "document.pointer*, originating.party?, receiving.party*"
>

<!ELEMENT originating.party (market.participant.info.pointer)>
<!ATTLIST originating.party
   %common.attrib;
>

<!ELEMENT receiving.party (market.participant.info.pointer)>
<!ATTLIST receiving.party
   %common.attrib;
>
```

```
<!ELEMENT confirm.component (%cbl.component.contents;)>
<!ATTLIST confirm.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
   confirming  CDATA    #REQUIRED
>

<!ELEMENT acknowledge.previous.component (%cbl.component.contents;)>
<!ATTLIST acknowledge.previous.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT request.for.quote.component (%cbl.component.contents;)>
<!ATTLIST request.for.quote.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT request.for.info.component (%cbl.component.contents;)>
<!ATTLIST request.for.info.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT response.component (%cbl.component.contents;)>
<!ATTLIST response.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT offer.to.sell.component (%cbl.component.contents;)>
<!ATTLIST offer.to.sell.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT catalogue.entry.component (%cbl.component.contents;)>
<!ATTLIST catalogue.entry.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT catalogue.component (%cbl.component.contents;)>
<!ATTLIST catalogue.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>
```

```
<!ELEMENT schedule.component (%cbl.component.contents;)>
<!ATTLIST schedule.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>


<!ELEMENT market.participant.info.component (%cbl.component.contents;)>
<!ATTLIST market.participant.info.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT shopping.cart.component (%cbl.component.contents;)>
<!ATTLIST shopping.cart.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT order.component (%cbl.component.contents;)>
<!ATTLIST order.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT invoice.component (%cbl.component.contents;)>
<!ATTLIST invoice.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT payment.notice.component (%cbl.component.contents;)>
<!ATTLIST payment.notice.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT shipment.notice.component (%cbl.component.contents;)>
<!ATTLIST shipment.notice.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT service.description.component (%cbl.component.contents;)>
<!ATTLIST service.description.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>
```

```
<!ELEMENT server.metadata.component (%cbl.component.contents;)>
<!ATTLIST server.metadata.component
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT market.description.component (%cbl.component.contents;)>
<!ATTLIST market.description.component
   %mimetype.attrib;
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT inventory.set.component (%cbl.component.contents;)>
<!ATTLIST inventory.set.component
   %mimetype.attrib;
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT escape.component (%cbl.component.contents;)>
<!ATTLIST escape.component
   %mimetype.attrib;
   %common.attrib;
   %component.attrib;
   %party.attrib;
>

<!ELEMENT we.are.here EMPTY>
<!ATTLIST we.are.here
   %common.attrib;
   %party.attrib;
>
```

```
<!-- infodesc.mod   Version:  0.6.2 -->
<!-- Purpose:  supply structure for all descriptions of information found in
   CBL XML documents -->
<!-- Terry Allen  29 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % infodesc.or.set
   "(info.description | info.description.set)"
>

<!ELEMENT info.description.set (%infodesc.or.set;,
   ((and, %infodesc.or.set;)*
   | (or, %infodesc.or.set;)*
   | (not, %infodesc.or.set;))
)>
<!ATTLIST info.description.set
   %common.attrib;
>

<!ELEMENT and EMPTY>
<!ATTLIST and
   %common.attrib;
>

<!ELEMENT or EMPTY>
<!ATTLIST or
   %common.attrib;
>

<!ELEMENT not EMPTY>
<!ATTLIST not
   %common.attrib;
>

<!ELEMENT info.description (xml.descriptor | nonxml.descriptor
   | urn.reference | regexp | range)>
<!ATTLIST info.description
   %common.attrib;
>

<!ELEMENT xml.descriptor (doctype, xml.descriptor.details)>
<!ATTLIST xml.descriptor
   %common.attrib;
>

<!ELEMENT nonxml.descriptor (%xll.or.urn;)>
<!ATTLIST nonxml.descriptor
   %common.attrib;
   cblpointer  CDATA    #FIXED      "outside"
>

<!ELEMENT regexp (#PCDATA)>
<!ATTLIST regexp
   %common.attrib;
>

<!ELEMENT doctype (dtd)>
```

```
<!ATTLIST doctype
   %common.attrib;
>

<!ELEMENT dtd EMPTY>
<!ATTLIST dtd
   systemid CDATA #IMPLIED
   publicid CDATA #IMPLIED
   %common.attrib;
>

<!ELEMENT xml.descriptor.details (xml.descriptor.context?,
      (xll.xptr.frag | xml.other.descriptor))>
<!ATTLIST xml.descriptor.details
   %common.attrib;
>

<!ELEMENT xml.descriptor.context EMPTY>
<!ATTLIST xml.descriptor.context
   %common.attrib;
   xll.link.traverse (none | all | all.recurse) "all.recurse"
>

<!ELEMENT xml.other.descriptor (#PCDATA)>
<!ATTLIST xml.other.descriptor
   %common.attrib;
   type  CDATA #REQUIRED
>

<!ELEMENT range (range.parameter, range.parameter, range.parameter*)>
<!ATTLIST range
   schema.name CDATA    #IMPLIED
   %common.attrib;
>

<!ELEMENT range.parameter (#PCDATA)>
<!ATTLIST range.parameter
   range.type (integer | decimal | nonnumeric) "decimal"
   schema.mapping CDATA #IMPLIED
   %common.attrib;
>
```

```
<!-- inventory.dtd   Version:  0.6 -->
<!-- Purpose:  provide inventory information -->
<!-- Terry Allen  25 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % proddesc SYSTEM "proddesc.mod">
%proddesc;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ELEMENT inventory.set (meta?, inventory.item+)>
<!ATTLIST inventory.set
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT inventory.item (market.participant.info.pointer+,
   inventory.id, product.description, stock.status?,
   quantity.in.stock?)>
<!ATTLIST inventory.item
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT inventory.id (%multilingual;)*>
<!ATTLIST inventory.id
   %common.attrib;
>
```

```
<!-- manifest.dtd Version:  0.7.1 -->
<!-- Purpose:  packing list for MIME message -->
<!-- Based on package.dtd from Terry Allen's
   "Unoptimized SGML-Bundle for MIME" proposal of February
   or March 1997,  "-//Palm Tree Books//DTD USB-Package v0.1//EN" -->
<!-- Terry Allen  4 Dec 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % manifest.ids.attrib
    "cidofpart CDATA #IMPLIED
    urnofentity CDATA #IMPLIED
    cidofentity CDATA #IMPLIED"
>

<!ELEMENT manifest ((mime.docentity.pointer | external.docentity.pointer),
   (mime.sgmldecl.pointer | external.sgmldecl.pointer)*,
   (mime.catalogue.pointer? | external.catalogue.pointer?),
   (mime.dtd.module.pointer | external.dtd.module.pointer)*,
   (mime.other.entity.pointer | external.other.entity.pointer)*)>

<!ELEMENT mime.docentity.pointer EMPTY>
<!ATTLIST mime.docentity.pointer
   %manifest.ids.attrib;
>

<!ELEMENT external.docentity.pointer EMPTY>
<!ATTLIST external.docentity.pointer
   %manifest.ids.attrib;
>

<!ELEMENT mime.sgmldecl.pointer EMPTY>
<!ATTLIST mime.sgmldecl.pointer
   %manifest.ids.attrib;
>

<!ELEMENT external.sgmldecl.pointer EMPTY>
<!ATTLIST external.sgmldecl.pointer
   %manifest.ids.attrib;
>

<!ELEMENT mime.catalogue.pointer EMPTY>
<!ATTLIST mime.catalogue.pointer
   %manifest.ids.attrib;
>

<!ELEMENT external.catalogue.pointer EMPTY>
<!ATTLIST external.catalogue.pointer
   %manifest.ids.attrib;
>

<!ELEMENT mime.dtd.module.pointer EMPTY>
<!ATTLIST mime.dtd.module.pointer
   %manifest.ids.attrib;
>

<!ELEMENT external.dtd.module.pointer EMPTY>
<!ATTLIST external.dtd.module.pointer
```

```
   %manifest.ids.attrib;
>

<!ELEMENT mime.other.entity.pointer EMPTY>
<!ATTLIST mime.other.entity.pointer
   %manifest.ids.attrib;
>

<!ELEMENT external.other.entity.pointer EMPTY>
<!ATTLIST external.other.entity.pointer
   %manifest.ids.attrib;
>
```

```
<!-- markdesc.dtd   Version:  0.6 -->
<!-- Purpose:  describe a marketplace -->
<!-- Terry Allen  25 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ENTITY % countrys SYSTEM "countrys.mod">
%countrys;

<!ENTITY % payment SYSTEM "paymento.mod">
%payment;

<!ENTITY % shipment SYSTEM "shipment.mod">
%shipment;

<!ENTITY % address SYSTEM "addresso.mod">
%address;

<!ENTITY % servprim SYSTEM "servprim.mod">
%servprim;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % transact SYSTEM "transact.mod">
%transact;

<!ELEMENT market.description (market.name,
   market.id?, market.operator+, market.registry+,
   market.terms.pointer)>
<!ATTLIST market.description
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT market.name (%multilingual;)*>
<!ATTLIST market.name
   %common.attrib;
>
```

```
<!ELEMENT market.id (%multilingual;)*>
<!ATTLIST market.id
   %common.attrib;
>

<!ELEMENT market.operator (market.operator.name,
   market.participant.info.pointer)>
<!ATTLIST market.operator
   %common.attrib;
>

<!ELEMENT market.operator.name (%multilingual;)*>
<!ATTLIST market.operator.name
   %common.attrib;
>

<!ELEMENT market.registry (market.registry.name,
   market.registry.id?, registry.role, service.location.pointer+)>
<!ATTLIST market.registry
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT market.registry.name (%multilingual;)*>
<!ATTLIST market.registry.name
   %common.attrib;
>

<!ELEMENT market.registry.id (%multilingual;)*>
<!ATTLIST market.registry.id
   %common.attrib;
>

<!ELEMENT registry.role EMPTY>
<!ATTLIST registry.role
   registry.contents (vendors | buyers | shippers) #REQUIRED
>
```

```
<!-- markpart.dtd  Version:  0.7.2 -->
<!-- Purpose:  groups market participant information -->
<!-- Terry Allen  5 Dec 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % countrys SYSTEM "countrys.mod">
%countrys;

<!ENTITY % addresso SYSTEM "addresso.mod">
%addresso;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % servprim SYSTEM "servprim.mod">
%servprim;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ELEMENT market.participant.info (meta?, (company.info | personal.info),
   service.list?)>
<!ATTLIST market.participant.info
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT company.info (company.name, previous.name*,
   (duns|duns4|mall.assigned), business.code*,
   start.date?, incorporated.in?,
   company.superentities?, company.subentities?,
   company.affiliation?, contact+)>
<!ATTLIST company.info
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT personal.info (personal.name, address.set+)>
<!ATTLIST personal.info
   %common.attrib;
>

<!ELEMENT business.code (naics.code | isic.code | jisx0403.code)>
<!ATTLIST business.code
   %common.attrib;
>

<!ELEMENT naics.code EMPTY>
```

```
<!ATTLIST naics.code
   %common.attrib;
   schema:naics   CDATA #REQUIRED
>

<!ELEMENT isic.code EMPTY>
<!ATTLIST isic.code
   %common.attrib;
   schema:isic CDATA    #REQUIRED
>

<!ELEMENT jisx0403.code EMPTY>
<!ATTLIST jisx0403.code
   %common.attrib;
   schema:jisx0403      CDATA #REQUIRED
>

<!ELEMENT company.name (%multilingual;)*>
<!ATTLIST company.name
   %common.attrib;
   schema:edifact CDATA #FIXED "organization"
>

<!ELEMENT previous.name (%multilingual;)*>
<!ATTLIST previous.name
   %common.attrib;
>

<!ELEMENT duns (#PCDATA)>
<!ATTLIST duns
   %common.attrib;
>

<!ELEMENT duns4 (#PCDATA)>
<!ATTLIST duns4
   %common.attrib;
>

<!ELEMENT mall.assigned (#PCDATA)>
<!ATTLIST mall.assigned
   %common.attrib;
>

<!ELEMENT naics (#PCDATA)>
<!ATTLIST naics
   %common.attrib;
>

<!ELEMENT isic (#PCDATA)>
<!ATTLIST isic
   %common.attrib;
>

<!ELEMENT fsc (#PCDATA)>
<!ATTLIST fsc
   %common.attrib;
>
```

```
<!ELEMENT start.date (date)>
<!ATTLIST start.date
   %common.attrib;
>

<!ELEMENT incorporated.in (country, country.subentity?)>
<!ATTLIST incorporated.in
   %common.attrib;
>

<!ELEMENT company.superentities (superentity+)>
<!ATTLIST company.superentities
   %common.attrib;
>

<!ELEMENT company.subentities (subentity+)>
<!ATTLIST company.subentities
   %common.attrib;
>

<!ELEMENT company.affiliation (#PCDATA)>
<!ATTLIST company.affiliation
   %common.attrib;
>

<!ELEMENT superentity (company.name, company.info.pointer?,
   company.superentities?)>
<!ATTLIST superentity
   %common.attrib;
>

<!ELEMENT subentity (company.name, company.info.pointer?,
   company.subentities?)>
<!ATTLIST subentity
   %common.attrib;
>

<!ELEMENT contact (contact.function*, personal.name*,
   language.understood*,
   occupation.title?, occupation.code?, address.set+)>
<!ATTLIST contact
   %common.attrib;
>

<!ELEMENT contact.function (%multilingual;)*>
<!ATTLIST contact.function
   %common.attrib;
>

<!ELEMENT language.understood EMPTY>
<!ATTLIST language.understood
   %lang.attrib.required;
>
```

```
<!-- modify.dtd   Version:  0.6.2 -->
<!-- Purpose:  group negotiation guide information -->
<!-- Terry Allen  29 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ELEMENT modification.set (meta?, modification.reason?,
   original.document.pointer,
   modified.document.pointer, modification*)>
<!ATTLIST modification.set
   %common.attrib;
   %party.attrib;
>

<!ELEMENT modification.reason EMPTY>
<!ATTLIST modification.reason
   %common.attrib;
   reason (product.substitution | price.change | quantity.change
      | tpa.change | self.explanatory) "self.explanatory"
>

<!ELEMENT original.document (document.pointer)>
<!ATTLIST original.document
   %common.attrib;
>

<!ELEMENT modified.document (document.pointer)>
<!ATTLIST modified.document
   %common.attrib;
>

<!ELEMENT modification (change.pointer.set | disagree.pointer.set
   | agree.pointer.set | prefer.pointer.set)*>
<!ATTLIST modification
   %common.attrib;
   %party.attrib;
>

<!ELEMENT change.pointer.set (change.pointer+)>
<!ATTLIST change.pointer.set
   %common.attrib;
   %party.attrib;
>

<!ELEMENT change.pointer (changed.from.pointer, changed.to.pointer)>
<!ATTLIST change.pointer
```

```
   %common.attrib;
>

<!ELEMENT changed.from.pointer (%xll.or.urn;)>
<!ATTLIST changed.from.pointer
   %common.attrib;
   cblpointer CDATA #FIXED "outside"
>

<!ELEMENT changed.to.pointer (%xll.or.urn;)>
<!ATTLIST changed.to.pointer
   %common.attrib;
   cblpointer CDATA #FIXED "outside"
>

<!ELEMENT disagree.pointer.set (disagree.pointer+)>
<!ATTLIST disagree.pointer.set
   %common.attrib;
   %party.attrib;
>

<!ELEMENT disagree.pointer (%xll.or.urn;)>
<!ATTLIST disagree.pointer
   %common.attrib;
   cblpointer CDATA #FIXED "outside"
>

<!ELEMENT prefer.pointer.set (prefer.pointer+)>
<!ATTLIST prefer.pointer.set
   %common.attrib;
   %party.attrib;
>

<!ELEMENT prefer.pointer (%xll.or.urn;)>
<!ATTLIST prefer.pointer
   %common.attrib;
   degree   CDATA "0"
   cblpointer CDATA #FIXED "outside"
>
```

```
<!-- ots.dtd  Version:  0.1 -->
<!-- Purpose:  define basic Offer To Sell -->
<!-- Terry Allen  2 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % proddesc SYSTEM "proddesc.mod">
%proddesc;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;


<!ELEMENT offer.to.sell (meta?,
   (market.participant.info.pointer | personal.info.pointer),
   xml.catalogue.pointer)>
<!ATTLIST offer.to.sell
   %common.attrib;
   %ttl.attrib;
>
```

```
<!-- paymento.mod   Version:  0.5 -->
<!-- Purpose:  group payment primitives -->
<!-- Terry Allen  9 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % issuer.name.attrib
   "issuer.name CDATA   #REQUIRED"
>

<!ELEMENT payment.method.set (payment.method+)>
<!ATTLIST payment.method.set
   %common.attrib;
>

<!ELEMENT payment.method (cash | credit.card | debit.card
   | check | eurocheck | bank.wire.transfer
   | postal.wire.transfer | ecash)>
<!ATTLIST payment.method
   %common.attrib;
>

<!ELEMENT cash EMPTY>
<!ATTLIST cash
   %currency.code.attrib;
>

<!ELEMENT credit.card EMPTY>
<!ATTLIST credit.card
   %issuer.name.attrib;
>

<!ELEMENT debit.card EMPTY>
<!ATTLIST debit.card
   %issuer.name.attrib;
>

<!ELEMENT check EMPTY>
<!ATTLIST check
   %currency.code.attrib;
   %country.name.attrib;
>

<!ELEMENT eurocheck EMPTY>
<!ATTLIST eurocheck
   %issuer.name.attrib;
>

<!ELEMENT bank.wire.transfer EMPTY>
<!ATTLIST bank.wire.transfer
   agent.name  CDATA    #REQUIRED
>

<!ELEMENT postal.wire.transfer EMPTY>
<!ATTLIST postal.wire.transfer
   %country.name.attrib;
>
```

```
<!ELEMENT ecash EMPTY>
<!ATTLIST ecash
   %currency.code.attrib;
   %issuer.name.attrib;
>

<!ELEMENT monetary.payment (currency.amount)>
<!ATTLIST monetary.payment
   %common.attrib;
>
```

```
<!-- paynoteo.dtd Version:  0.5 -->
<!-- Purpose:  describe a payment notice-->
<!-- Terry Allen  9 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ENTITY % countrys SYSTEM "countrys.mod">
%countrys;

<!ENTITY % payment SYSTEM "paymento.mod">
%payment;

<!ENTITY % shipment SYSTEM "shipment.mod">
%shipment;

<!ENTITY % address SYSTEM "addresso.mod">
%address;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ELEMENT payment.notice (meta?, payment.method, monetary.payment,
   payor, payee, date.and.time)>
<!ATTLIST payment.notice
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT payor (market.participant.info.pointer | personal.info.pointer)>
<!ATTLIST payor
   %common.attrib;
>

<!ELEMENT payee (market.participant.info.pointer | personal.info.pointer)>
<!ATTLIST payee
   %common.attrib;
>
```

```
<!-- price.mod   Version:  0.5 -->
<!-- Purpose:  group address information primitives -->
<!-- Terry Allen  8 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ELEMENT price.group (base.price?, price.with.tax?, discount.set?,
   discounted.price?, price.adjust*,
   adjusted.price?, ship.charge.set?, total.price?)>
<!ATTLIST price.group
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT base.price (currency.amount)>
<!ATTLIST base.price
   %common.attrib;
   %ttl.attrib;
   tax.included   (yes | no)  "no"
>

<!ELEMENT price.with.tax (currency.amount)>
<!ATTLIST price.with.tax
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT discount.set (discount+)>
<!ATTLIST discount.set
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT discount (discount.name?, discount.id?,
   discount.rate?, discount.amount?)>
<!ATTLIST discount
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT discount.name (%multilingual;)*>
<!ATTLIST discount.name
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT discount.id (%multilingual;)*>
<!ATTLIST discount.id
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT discount.rate (rate)>
<!ATTLIST discount.rate
   %common.attrib;
   %ttl.attrib;
>
```

```
<!ELEMENT rate (#PCDATA)>
<!ATTLIST rate
   %common.attrib;
   %ttl.attrib;
   rate.basis (percent|other) "percent"
>

<!ELEMENT discounted.price (currency.amount)>
<!ATTLIST discounted.price
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT discount.amount (currency.amount)>
<!ATTLIST discount.amount
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT price.adjust (tax.set?, other.charge.set?)>
<!ATTLIST price.adjust
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT adjusted.price (currency.amount)>
<!ATTLIST adjusted.price
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT tax.set (tax+)>
<!ATTLIST tax.set
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT tax (tax.name?, tax.id?, tax.rate?, tax.amount?)>
<!ATTLIST tax
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT tax.name (%multilingual;)*>
<!ATTLIST tax.name
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT tax.id (%multilingual;)*>
<!ATTLIST tax.id
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT tax.rate (rate)>
<!ATTLIST tax.rate
```

```
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT tax.amount (currency.amount)>
<!ATTLIST tax.amount
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT other.charge.set (other.charge+)>
<!ATTLIST other.charge.set
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT other.charge (other.charge.name?,
   other.charge.id?, other.charge.rate?, other.charge.amount)>
<!ATTLIST other.charge
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT other.charge.name (%multilingual;)*>
<!ATTLIST other.charge.name
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT other.charge.id (%multilingual;)*>
<!ATTLIST other.charge.id
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT other.charge.rate (rate)>
<!ATTLIST other.charge.rate
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT other.charge.amount (currency.amount)>
<!ATTLIST other.charge.amount
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT ship.charge.set (ship.charge+)>
<!ATTLIST ship.charge.set
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT ship.charge (ship.charge.name?,
   ship.charge.id?, ship.charge.amount)>
<!ATTLIST ship.charge
   %common.attrib;
```

```
   %ttl.attrib;
>

<!ELEMENT ship.charge.name (%multilingual;)*>
<!ATTLIST ship.charge.name
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT ship.charge.id (%multilingual;)*>
<!ATTLIST ship.charge.id
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT ship.charge.amount (currency.amount)>
<!ATTLIST ship.charge.amount
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT total.price (currency.amount)>
<!ATTLIST total.price
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT total.discount (currency.amount)>
<!ATTLIST total.discount
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT total.adjustment (currency.amount)>
<!ATTLIST total.adjustment
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT grand.total.price (currency.amount)>
<!ATTLIST grand.total.price
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT price.range (minimum?, maximum?)>
<!ATTLIST price.range
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT minimum (currency.amount)>
<!ATTLIST minimum
   %common.attrib;
   %ttl.attrib;
>
```

```
<!ELEMENT maximum (currency.amount)>
<!ATTLIST maximum
   %common.attrib;
   %ttl.attrib;
>
```

```
<!-- proddesc.mod   Version:  0.6 -->
<!-- Purpose:  provide simplest product description chunk -->
<!-- Terry Allen  26 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ELEMENT product.description (product.name?, product.id*,
   product.line?, brand?, catalogue.category?, keyword.set,
   material.set?, color.set?, size.set*, warranty?,
   text.description)>
<!ATTLIST product.description
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT short.product.description (product.name?, product.id*,
   product.line?, brand?, catalogue.category?,
   material.choice?, color.choice?, size.choice?)>
<!ATTLIST short.product.description
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT product.name (%multilingual;)*>
<!ATTLIST product.name
   %common.attrib;
>

<!ELEMENT product.id (%multilingual;)*>
<!ATTLIST product.id
   %common.attrib;
   type (upc | other)   #IMPLIED
   assigned.by
   (manufacturer | distributor | vendor | nother) #IMPLIED
>

<!ELEMENT product.line (%multilingual;)*>
<!ATTLIST product.line
   %common.attrib;
>

<!ELEMENT brand (%multilingual;)*>
<!ATTLIST brand
   %common.attrib;
>

<!ELEMENT catalogue.category (taxon.pointer+)>
<!ATTLIST catalogue.category
   %common.attrib;
>

<!ELEMENT keyword.set (keyword+)>
<!ATTLIST keyword.set
   %common.attrib;
>

<!ELEMENT keyword (#PCDATA)>
<!ATTLIST keyword
```

```
   %common.attrib;
   taxonomy CDATA #IMPLIED
>

<!ELEMENT text.description (%multilingual;)*>
<!ATTLIST text.description
   %common.attrib;
>

<!ELEMENT material.set (material+)>
<!ATTLIST material.set
   %common.attrib;
   conjunction (combination | alternatives)     "alternatives"
>

<!ELEMENT material.choice (material)>
<!ATTLIST material.choice
   %common.attrib;
>

<!ELEMENT material (%multilingual;)*>
<!ATTLIST material
   %common.attrib;
>

<!ELEMENT color.set (color+)>
<!ATTLIST color.set
   %common.attrib;
   conjunction (combination | alternatives)     "alternatives"
>

<!ELEMENT color.choice (color)>
<!ATTLIST color.choice
   %common.attrib;
>

<!ELEMENT color (%multilingual;)*>
<!ATTLIST color
   %common.attrib;
>

<!ELEMENT size.set (size.qualifier?, size+)>
<!ATTLIST size.set
   %common.attrib;
   units CDATA #REQUIRED
>

<!ELEMENT size.choice (size)>
<!ATTLIST size.choice
   %common.attrib;
   units CDATA #REQUIRED
>

<!ELEMENT size.qualifier (%multilingual;)*>
<!ATTLIST size.qualifier
   %common.attrib;
>
```

```
<!ELEMENT size (%multilingual;)*>
<!ATTLIST size
   %common.attrib;
>

<!ELEMENT warranty (%multilingual;)*>
<!ATTLIST warranty
   %common.attrib;
>

<!ELEMENT stock.status EMPTY>
<!ATTLIST stock.status
   %common.attrib;
   in.stock (yes | no)  "yes"
   %ttl.attrib;
>

<!ELEMENT quantity.in.stock (#PCDATA)>
<!ATTLIST quantity.in.stock
   %common.attrib;
>
```

```
<!-- response.dtd Version:  0.23 -->
<!-- Purpose:  define response to query.dtd -->
<!-- Terry Allen  26 Oct 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ELEMENT response (meta?, input.pointer,
   (error.response | success.response | delay.response)?,
   record.pointer*)>
<!ATTLIST response
   %common.attrib;
>

<!ELEMENT input.pointer (%xll.or.urn;)>
<!ATTLIST input.pointer
   %common.attrib;
   %xll.exlink.attrib;
   cblpointer  CDATA    #FIXED      "content"
   target.dtd  CDATA    #FIXED "infodesc.dtd"
>

<!ELEMENT error.response EMPTY>
<!ATTLIST error.response
   %common.attrib;
   type  (unspecified | unauthorized | returntype.mismatch)
      #REQUIRED
>

<!ELEMENT success.response EMPTY>
<!ATTLIST success.response
   %common.attrib;
>

<!ELEMENT delay.response (date.and.time | duration)>
<!ATTLIST delay.response
   %common.attrib;
>

<!ELEMENT record.pointer (%xll.or.urn;)>
<!ATTLIST record.pointer
   %common.attrib;
   %xll.exlink.attrib;
   cblpointer  CDATA    #FIXED      "content"
>
```

```
<!-- rfq.dtd  Version:  0.1 -->
<!-- Purpose:  define basic Request for Quote -->
<!-- Terry Allen  2 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % proddesc SYSTEM "proddesc.mod">
%proddesc;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;


<!ELEMENT request.for.quote (meta?,
   (market.participant.info.pointer | personal.info.pointer),
   desideratum+)>
<!ATTLIST request.for.quote
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT desideratum (product.description, price.range?)>
<!ATTLIST desideratum
   %common.attrib;
   %ttl.attrib;
>
```

```
<!-- semantic.dtd   Version:  0.2 -->
<!-- Purpose:  group semantics for DTDs -->
<!-- Terry Allen  15 Oct 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ELEMENT fpi (name, ((commonatts?, element*) | code.list))>
<!ATTLIST fpi
   %common.attrib;
>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name
   %common.attrib;
>

<!ELEMENT commonatts (attribute+)>
<!ATTLIST commonatts
   %common.attrib;
>

<!ELEMENT element (name, meaning, attribute*)>
<!ATTLIST element
   %common.attrib;
>

<!ELEMENT meaning (%multilingual;)*>
<!ATTLIST meaning
   %common.attrib;
>

<!ELEMENT attribute (name, meaning, def.list.item*,
   (default | fixed | implied | required))>
<!ATTLIST attribute
   %common.attrib;
>

<!ELEMENT def.list.item (name, meaning)>
<!ATTLIST def.list.item
   %common.attrib;
>

<!ELEMENT default (name)>
<!ATTLIST default
   %common.attrib;
>

<!ELEMENT fixed ((name?, meaning)+)>
<!ATTLIST fixed
   %common.attrib;
>

<!ELEMENT implied (name, meaning?)>
<!ATTLIST implied
   %common.attrib;
```

```
>

<!ELEMENT required EMPTY>
<!ATTLIST required
   %common.attrib;
>

<!ELEMENT code.list (code.list.name, specification, (name, meaning)+)>
<!ATTLIST code.list
   %common.attrib;
>

<!ELEMENT code.list.name (%multilingual;)*>
<!ATTLIST code.list.name
   %common.attrib;
>

<!ELEMENT specification (%multilingual;)*>
<!ATTLIST specification
   %common.attrib;
>
```

```
<!-- servdesc.dtd  Version:  0.5 -->
<!-- Purpose:  describe service -->
<!-- Terry Allen  8 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % countrys SYSTEM "countrys.mod">
%countrys;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % addresso SYSTEM "addresso.mod">
%addresso;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % servprim SYSTEM "servprim.mod">
%servprim;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ELEMENT service.description (meta?, service)>
<!ATTLIST service.list
   %common.attrib;
>
```

```
<!-- servmeta.dtd   Version:  0.5 -->
<!-- Purpose:  describe a server's metainformation about a document -->
<!-- Terry Allen  8 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ELEMENT server.metadata (description?, urn?, url?, version?, time.created?,
   time.last.modified?, altrep*)>
<!ATTLIST server.metadata
   %common.attrib;
>
```

```
<!-- shopcart.dtd Version:  0.5 -->
<!-- Purpose:  describe a shopping cart -->
<!-- Terry Allen  9 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % ttl SYSTEM "ttlattri.mod">
%ttl;

<!ENTITY % datetime SYSTEM "datetime.mod">
%datetime;

<!ENTITY % currency SYSTEM "currency.mod">
%currency;

<!ENTITY % proddesc SYSTEM "proddesc.mod">
%proddesc;

<!ENTITY % price SYSTEM "price.mod">
%price;

<!ENTITY % countrys SYSTEM "countrys.mod">
%countrys;

<!ENTITY % payment SYSTEM "paymento.mod">
%payment;

<!ENTITY % shipment SYSTEM "shipment.mod">
%shipment;

<!ENTITY % address SYSTEM "addresso.mod">
%address;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ENTITY % meta SYSTEM "meta.mod">
%meta;

<!ELEMENT shopping.cart (meta?, session.id,
   market.participant.info.pointer, item.selected*,
   total.discount?, total.adjustment?, grand.total.price?)>
<!ATTLIST shopping.cart
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT session.id (%multilingual;)*>
<!ATTLIST session.id
   %common.attrib;
   %party.attrib;
>

<!ELEMENT item.selected (short.product.description,
   vendor.name?, quantity.desired, price.group,
```

```
   item.subtotal?)>
<!ATTLIST item.selected
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT vendor.name (#PCDATA)>
<!ATTLIST vendor.name
   %common.attrib;
>

<!ELEMENT quantity.desired (#PCDATA)>
<!ATTLIST quantity.desired
   %common.attrib;
>

<!ELEMENT item.subtotal (#PCDATA)>
<!ATTLIST item.subtotal
   %common.attrib;
>
```

```
<!-- taxonomy.dtd   Version:  0.7.2 -->
<!-- Purpose:  define taxonomy structure -->
<!-- Terry Allen  4 Dec 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->

<!ENTITY % common SYSTEM "commatts.mod">
%common;

<!ENTITY % pointers SYSTEM "pointers.mod">
%pointers;

<!ELEMENT taxon (taxon.name, taxon.urn,
      taxon.info?, taxon.parent.pointer*,
      (taxon.child.pointer | taxon)*)>
<!ATTLIST taxon
   %common.attrib;
>

<!ELEMENT taxon.name (%multilingual;)*>
<!ATTLIST taxon.name
   %common.attrib;
>

<!ELEMENT taxon.urn (#PCDATA)>
<!ATTLIST taxon.urn
   %common.attrib;
>

<!ELEMENT taxon.info (%multilingual;)*>
<!ATTLIST taxon.info
   %common.attrib;
>
```

```
<!-- transact.mod Version:  0.6 -->
<!-- Purpose:  describe a transaction -->
<!-- Terry Allen  25 Nov 1997 -->
<!-- Copyright 1997 CNgroup, Inc. -->


<!ELEMENT transaction.set (transaction.set+ | transaction+)>
<!ATTLIST transaction.set
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT transaction (transaction.id+, tpa.pointer*,
   (market.participant.info.pointer | personal.info.pointer)+,
   exchange.description, exchange.description,
   exchange.description*)>
<!ATTLIST transaction
   %common.attrib;
   %ttl.attrib;
>

<!ELEMENT exchange.description ((commerce.item,
   shipment.coordinates.set?, payment.method?)+)>
<!ATTLIST exchange.description
   from.party  CDATA    #REQUIRED
   to.party CDATA #REQUIRED
>

<!ELEMENT commerce.item ((
   (retail.catalogue.item.ordered, info.description.set?)
   | monetary.payment),
   total.discount?, total.adjustment?,
   grand.total.price?, shipto.address?, billto.address?)>
<!ATTLIST commerce.item
   %common.attrib;
>

<!ELEMENT transaction.id (%multilingual;)*>
<!ATTLIST transaction.id
   %common.attrib;
   %party.attrib;
>

<!ELEMENT tpa.pointer (%xll.or.urn;)>
<!ATTLIST tpa.pointer
   %common.attrib;
   %xll.exlink.attrib;
   cblpointer  CDATA    #FIXED      "content"
>

<!ELEMENT retail.catalogue.item.ordered (catalogue.entry.pointer,
   sku?, quantity.ordered, price.group?)>
<!ATTLIST retail.catalogue.item.ordered
   %common.attrib;
   %ttl.attrib;
>
```

```
<!ELEMENT quantity.ordered (#PCDATA)>
<!ATTLIST quantity.ordered
   %common.attrib;
>
```